UNIVERSIDADE FEDERAL DE SANTA CATARINA
CAMPUS TRINDADE
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
CURSO DE CIÊNCIAS DA COMPUTAÇÃO

GABRIEL RAMILDES FERREIRA

INTLIN: UM SISTEMA PARA AUXÍLIO NO ESTUDO DE LÍNGUA ESTRANGEIRAS

Florianópolis, SC Brasil
2022

GABRIEL RAMILDES FERREIRA

INTLIN: UM SISTEMA PARA AUXÍLIO NO ESTUDO DE LÍNGUA ESTRANGEIRAS

Trabalho Conclusão do Curso de Graduação em Ciências da Computação do Centro de Informática e estatística da Universidade Federal de Santa Catarina como requisito para a obtenção do título de Bacharel em Ciências da Computação.
Orientador: Prof. Raul Sidnei Wazlawick, Dr..

Florianópolis, SC Brasil

2022

Ficha de identificação da obra

Ferreira, Gabriel Ramildes Ferreira
Intlin : um sistema para auxílio no estudo de línguas estrangeiras / Gabriel Ramildes Ferreira Ferreira ; orientador, Raul Sidnei Wazlawick, 2022.
124 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, , Graduação em , Florianópolis, 2022.

Inclui referências.

1. . 2. Aprendizado de idiomas. 3. Desenvolvimento de software. 4. Repetição espaçada. 5. Web Scraping. I. Wazlawick, Raul Sidnei. II. Universidade Federal de Santa Catarina. Graduação em . III. Título.

Gabriel Ramildes Ferreira

INTLIN: UM SISTEMA PARA AUXÍLIO NO ESTUDO DE LÍNGUA ESTRANGEIRAS

Este Trabalho Conclusão de Curso foi julgado adequado para obtenção do Título de Bacharel e aprovado em sua forma final pelo Curso Ciências da computação

Florianópolis, 26 de Julho de 2022.

_____

Prof. Jean Everson Martina
Coordenador do Curso

Banca Examinadora:

_____

Prof.(a) Raul Sidnei Wazlawick
Orientador(a)
Instituição UFSC

_____

Prof.(a) Carina Friedrich Dorneles
Avaliador(a)
Instituição UFSC

_____

Prof.(a) Christiane Anneliese Gresse Von Wangenheim
Avaliador(a)
Instituição UFSC

Dedico este trabalho a todas as pessoas que eu amo

AGRADECIMENTOS

Agradeço a todos aqueles que me apoiaram durante a graduação, em especial aos meus pais, que muito se esforçaram para que eu chegasse aqui. Agradeço a todos os professores que foram importantes na minha formação profissional e pessoal. E aos meus amigos que estiveram comigo até o fim, meus sinceros agradecimentos.

RESUMO

Dentre as diversas formas possíveis para aprender um novo idioma, um método de estudo muito comum por estudantes autodidatas é o de exposição massiva ao idioma estudado, por meio do consumo de mídias nativas e autênticas, concorrentemente pesquisando vocabulários, conforme a necessidade. Muitas vezes, também, utiliza-se flashcards em um sistema de repetição espaçadas (SRS) para tratar de revisões. Em termos de eficiência, o maior problema desse método, quando utilizado apenas recursos gratuitos, é que o estudante deve administrar diversas ferramentas. Além de ter seu conteúdo aberto, deve procurar algum dicionário externo, e para criação de *flashcards*, além de necessário instalar algum sistema de repetição espaçada, é necessário anotar em algum lugar as informações para os cartões que se deseja criar, ou até mesmo, parar tudo que está fazendo para criá-los. Devido a esse processo bastante trabalhoso e manual, este trabalho desenvolveu um aplicativo desktop que integre três módulos principais em um único serviço: (i) Módulo de mídia, (ii) Módulo de dicionários e (iii) Módulo SRS. Também permitindo a criação rápida de flashcards, tendo o objetivo de otimizar o tempo de estudo e automatizar o processo de criação de cartões. Também desenvolveu-se um bot para coletar dados da web de definições de dicionários para alguns idiomas, tais dados foram usados pelo módulo de dicionários do sistema. Para tal desenvolvimento, utilizou-se a linguagem de programação Java junto com a plataforma JavaFX. Ao final do trabalho, chegou-se em um protótipo funcional do aplicativo especificado anteriormente, um web scraper completo para extração de dados dos dicionários, bem como resultados de uma avaliação do sistema na prática.

Palavras-chave: Aprendizado de idiomas. Desenvolvimento de software. Repetição espaçada. JavaFX. Web Scraping.

LISTA DE ILUSTRAÇÕES

Índice de tabelas

LISTA DE ABREVIATURAS E SIGLAS

SRS – Spaced Repetition System

MVC – Model-View-Controller

GUI – Graphical User Interface

API – Application Programming Interface

TDD – Test Driven Development

SUS – System Usabilty Scale

Sumário

1 INTRODUÇÃO

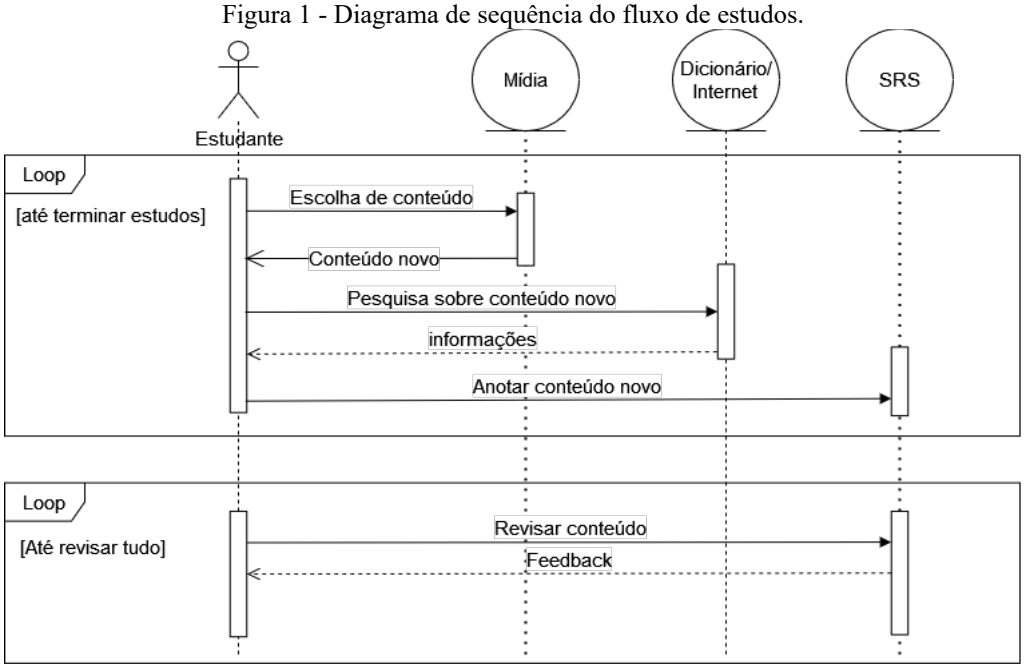
Dentro de métodos de aprendizagem de idiomas, abordagens que tem como principal princípio a exposição massiva a *input*, preferencialmente de conteúdos de interesse pessoal do estudante são muito populares entre estudantes. A teoria de aprendizagem via *input* foi levantada na acadêmia pela primeira vez pelo Professor da Universidade do Sul da Califórnia, Stephen Krashen (Kreshen, 1985). Apesar de bastante difundida e ter gerado uma mudança de paradigmas de como idiomas são ensinados em sala de aula (Patsy & Spada, 2006) (Dayan, 2015), sua hipótese do *input* (*Input Hypothesis*, no original) é uma das hipóteses de aquisição de segunda língua mais controversas (Brown, 2006) no meio acadêmico. Dayan Liu afirma que a hipótese de Krashen é vaga, não havendo definições exatas de seus conceitos (Dayan, 2015), outra crítica seria que as suas teses seriam impossíveis de um teste empírico (McLaughlin, 1987). Mesmo com tais objeções, o trabalho de Krashen é muito influente e de bastante importância em pesquisas da linguística, tendo surgido adaptações de sua hipótese que tentam suprir eventuais deficiências teóricas (Dayan, 2015), bem como diversas pesquisas e estudos em cima de sua hipótese. Além de métodos de aprendizagem baseados em *input* terem sido amplamente difundidos, há pesquisas indicando que estudantes, que já tenham um certo nível, conseguem um progresso considerável por exposição a uma língua alvo, sendo recomendado como um método suplementar para estudantes intermediários ou avançados (Patsy & Spada, 2006).

Como mencionado, a hipótese de Krashen foi muito difundida e desde de quando foi proposta, surgiu e adaptaram-se diversos métodos seguindo essa linha como base. Dentre internautas, é muito comum o sentimento que exposição massiva ao idioma por meio de *input* com conteúdos de interesse pessoal seria a forma mais eficiente de aprender tal língua. Acima disso, frequentemente são usados sistemas de repetição espaçada, em que o estudante cria *flashcards* (seja de vocabulário, sentenças, diálogos, ou qualquer outro a gosto) que são cartões que serão apresentados para revisão em períodos de tempos variantes, feito de modo a tomar proveito do *spacing effect*, fenômeno psicológico observado pela primeira vez por Hermann Ebbinghaus (1885) [2]. Tais cartões possuem uma frente e um verso, a frente possui o conteúdo principal a ser estudado, e o verso possui informações para que o estudante verifique se realmente acertou o conteúdo do cartão, ou o relembre em caso de erro. Há diversos benefícios da repetição espaçada no aprendizado, dentre outros, ela aprimora a habilidade de lembrança do estudante (Damaris, 2014).

Deste modo, um plano de estudo baseado no especificado anteriormente, usualmente segue o seguinte fluxo descrito na Figura 1. (1) O estudante escolhe uma mídia de seu interesse e em nível apropriado, (2) ele consome dada mídia, atento a estruturas e vocabulários não conhecidos, (3)

realiza uma pesquisa sobre os pontos desconhecidos encontrados e os anota (4) cria-se *flashcards* para revisão deste novo conteúdo. (5) O estudante realiza a revisão de cartões agendados para o dia.

Figura 1 - Diagrama de sequência do fluxo de estudos.



Fonte: produção do autor

1.1 PROBLEMÁTICA

Considerando o plano de estudos especificado na seção anterior, um estudante desejará utilizar algum recurso para tal modo de estudo, contudo a escassez de um sistema único que agrupe, gratuitamente, todos os recursos necessários para o seguimento de tal plano de estudo se torna evidente.

Por meio da internet, o trabalho de encontrar alguma mídia é trivial. O acesso a informações sobre vocabulários e estruturas do idioma também são amplamente acessíveis. Outrossim, existem diversos softwares de repetição espaçadas gratuitos que podem ser utilizados para o fim de anotações e revisões, sendo o *Anki* um dos mais populares. Desta forma, apenas com recursos gratuitos já seria possível realizar estudos deste tipo, porém tais recursos são todos desconexos entre si, necessitando tanto da administração de múltiplos recursos, como que haja uma pausa do consumo da mídia em estudo para realização de consultas e anotações, é possível imaginar como a constante remoção de foco do conteúdo estudado pode tornar o processo de estudo pouco fatigoso e menos engajante. Ademais, sistemas de repetição espaçadas (ou SRS) costumam permitir inserção de mídias nos cartões, servindo como um complemento áudio ou visual aos textos nos cartões ou, até mesmo como o conteúdo em si do cartão, isso exige do estudante realizar capturas de

telas manualmente, cortar clipes de áudio ou vídeo com algum outro recurso a parte, para que então possa inserir esse pedaço de mídia em seu cartão.

Assim, o problema principal se dá pela dificuldade de encontrar um sistema que integre diversos recursos em um único sistema, para que o processo de estudo seja mais fluído e prazeroso.

1.2 OBJETIVOS

Este trabalho propõe desenvolver um sistema para auxiliar o estudo de idiomas estrangeiros por método baseado em *input*, bem como um web scraper para coleta de dados de dicionários, gerando como subproduto um grande conjunto de dados de dicionários, que ficará disponível em domínio público.

A plataforma desktop para auxiliar no aprendizado de idiomas estrangeiros integra em um único pacote: (i) um sistema de dicionários de múltiplos idiomas, (ii) um sistema de repetição espaçada próprio e (iii) um módulo de recursos multimídia, em qual é possível abrir mídias para estudo a gosto. Visando, assim, agilizar o processo de criação de flashcards e manter todos os recursos em um único pacote para que seja possível manter o foco na mídia principal estudada. Tendo como especificações ser um **sistema para uso gratuito**, que permita a **criação de flashcards** tanto manualmente como automaticamente a partir de recursos do sistema, possua um **sistema interno de dicionário**, bem como permitir que o usuário **abra mídias de conteúdos de interesse próprio**. Tal sistema será nomeado de INTLIN, a junção de integrated (integrado, do inglês) com língua, do português.

Para o sistema de dicionários, deseja-se montar um modulo completamente *offline*, para isso será necessário conseguir um conjunto de dados que possam ser utilizados pelo sistema, sem necessidade de consumir uma API *online*. Deste modo, além do principal aplicativo proposto, desenvolver-se-á um *bot* em para extrair dados de dicionário, a partir do site *Wiktionary*. Esse robô será desenvolvido em Python utilizando o framework para *web scraping*, Scrapy. Esse web scraper será nomeado de DSPD, acrônimo para *Dictionary Spider.*

1.2.1 Objetivos específicos

- O1. Obter conjunto de dados para construção de dicionários. Tais dados devem ser permitidos para uso livre, isto é, sua licença não exija tributo aos autores original dos dados.
  - O1.1. Construir um *Web Scraper* para coletar dados do *Wiktionary* de alguns idiomas.
- O2. Desenvolver um sistema de dicionários que utilize os dados coletados.

- O3. Desenvolver um módulo de mídias, o qual o usuário pode abrir conteúdos de seu interesse na língua alvo, para que então possa estudá-lo.
- O4. Desenvolver um módulo de sistema de repetição espaçada, no qual são inseridos flashcards para revisão.

2 TRABALHOS RELACIONADOS

No mercado, existem sistemas similares ao descrito, mas não abrangendo todas as especificações colocadas na seção anterior: sistema para uso gratuito, criação de flashcards (manualmente e automaticamente), sistema interno de dicionário e possibilitar abrir mídias de conteúdos de interesse próprio para estudo. Sendo o fato de outros sistemas não serem completamente gratuitos o maior problema comum identificado.

Existem também diversos outros aplicativos gratuitos de bastante sucesso no mercado, tal como Duolingo, entretanto a proposta desse aplicativo é fundamentalmente diferente do proposto pelo INTLIN.

Os sistemas escolhidos para comparação são apenas aqueles que possuem a abordagem de estudo especificado: Utilizar mídias autênticas de interesse do estudante e ter um sistema de revisões baseados em repetição espaçada. Por isso, o sistema Duolingo ficará de fora da comparação realizada em 2.5 Comparativo geral. Mesmo assim, o mesmo recebeu uma seção própria devido a sua popularidade, bem como para esclarecer o porquê de aplicativos desse estilo serem diferentes da proposta do INTLIN.

2.1 Duolingo

Duolingo é um sistema de estudos de idioma de grande sucesso, sendo possivelmente o aplicativo mais popular de aprendizado de idiomas, tendo mais de 300 milhões de usuários, de acordo com o portal oficial (Duolingo).

Apesar de seu enorme sucesso, sua proposta é bastante diferente do que se deseja alcançar com o INTLIN. O Duolingo segue um sistema de lições ordenadas por tópicos que vão sendo desbloqueados progressivamente, similar a um sistema de níveis em um videogame. As lições são aglomerados de exercícios constituídos primariamente de tradução de sentenças, tais frases começam simples e progressivamente aumentam em dificuldade. O principal a se observar é que as sentenças dos exercícios são acompanhadas de pouco contexto, que na sua ausência pode acarretar na perda de entendimento de certas nuanças, ou até mesmo ambiguidade (Agnieszka & Elżbieta, 2018) (Yuta, 2022).

Com isso, apesar de todo o sucesso do aplicativo, o Duolingo tem uma proposta diferente do INTLIN e dos demais trabalhos a serem tratados neste capítulo, os quais têm como foco principal o estudo da língua por meio de mídias autênticas e naturalmente munidas de contexto, em oposição ao sistema de exercícios progressivos do Duolingo.

2.2 FluentU

O FluentU é uma plataforma para aprendizado de línguas, que disponibiliza uma extensa biblioteca de vídeos em diferentes níveis de dificuldade, sendo possível pesquisar definições de palavras vistas nas lições, flashcards são criados automaticamente a partir de lições estudadas, também sendo possível criá-los manualmente, conforme o estudante necessitar. Apesar de permitir criar flashcards de modo manual, é possível apenas criar sobre vocabulários, não dando muito liberdade ao estudante para criar os cartões que desejar. Outrossim, não é possível estudar qualquer mídia de interesse do estudante, ficando atado apenas a seleção de vídeos do serviço.

Assim, esta plataforma contempla as principais especificações do INTLIN, além de possuir vídeos autorais para estudo. Contudo, por não ser gratuita, não é uma ferramenta muito acessível. Desta forma, sua diferença com o INTLIN torna-se evidente. Enquanto o FluentU já é um serviço bastante robusto, ele não permite o mesmo grau de liberdade na escolha de conteúdos e na criação de flashcards que o INTLIN ambiciona, além de ser um serviço pago.

2.3 Yabla

Este serviço é bastante similar ao FluentU, possuindo uma biblioteca extensa de vídeos e um sistema de repetição espaçada para revisão do conteúdo. Também é interessante ressaltar que o sistema é bem avaliado por diversos órgãos públicos de renome, tais como universidade de Michigan e departamento de estado dos Estados Unidos (Yabla).

Comparando-se ao FluentU, esse serviço não permite criação de flashcards manualmente, e possui uma gama menor de idiomas disponíveis. Apesar de ainda ser pago, tem a vantagem de ser mais barato. Também não é possível importar nenhum tipo de mídia própria para o sistema, estando o estudante limitado a biblioteca nativa do sistema.

2.4 LingQ

LingQ, pronunciado como “link”, é um serviço muito próximo do proposto pelo INTLIN, com diversos extras e algumas desvantagens. Neste serviço, existem diversas lições, em formato de texto, áudio e/ou vídeo, disponibilizadas pela pela plataforma. Naturalmente, é possível pesquisar vocabulários enquanto se estuda uma lição, e então salvá-los para revisão em um sistema de SRS, e para palavras salvas manualmente, flashcards adicionais com elas utilizadas em frases são criados

automaticamente. Além das lições do sistema, é possível importar algum conteúdo de interesse próprio, podendo ser textos, áudios ou vídeos, de acordo com o gosto do estudante.

Além do mencionado, o LingQ vai muito além, possuindo recursos de comunidade para comunicação entre estudantes, como fóruns e intercambio de escritas. Também disponibiliza sistema para agendamento de tutorias com professores que utilizam a plataforma. Assim, o LingQ vai além de um sistema que ajuda o estudante a evoluir a compreensão de um idioma, e disponibiliza recursos para o estudante melhorar seu *output*, i.e. escrita e fala.

Apesar de um ótimo serviço, o uso do LingQ possui uma assinatura *Premium*, que gera limitações para usuários gratuitos, havendo um limite para quantidade de flashcards que podem ser criados e de lições importadas, deixando o usuário preso apenas com lições do sistema e com números limitados de flashcards, após exceder os limites.

2.5 Comparativo geral

As principais plataformas similares ao INTLIN foram discutidas nas seções anteriores. A Tabela 1, a seguir, faz uma comparação do que se propõe desenvolver com estas plataformas, de acordo com os pontos propostos para o trabalho.

Tabela 1 - Comparação deste trabalho X sistemas similares

X	FluentU	Yabla	Lingq	INTLIN
Sistema de dicionário interno	✔	✔	✔	✔
Flashcards automáticos e manuais	✔	✘	✔	✔
Abrir conteúdo de interesse próprio	✘	✘	✔	✔
100% Gratuito	✘	✘	✘	✔
Conteúdos autorais para estudar	✔	✔	✔	✘

Os pontos em amarelo na comparação indicam que a plataforma contempla a funcionalidade, mas com algumas ressalvas. O FluentU não permite uma flexibilidade muito grande na criação dos seus flashcards, como já discutido em sua seção. O LingQ também só permite criar flashcards de vocabulários vistos nas lições, não permitindo que o usuário personalize seus cartões de acordo com suas necessidades.

Além disso, as outras três plataformas da comparação disponibilizam conteúdos próprios em diferentes níveis, o que não é possível oferecer no INTLIN, devido ao fato de ser uma aplicação gratuita e não haver verba para elaboração profissional de conteúdos autorais, sendo delegado ao usuário encontrar algum conteúdo apropriado de seu interesse para estudo.

Dessarte, o INTLIN **não é** uma plataforma que *ensina* um idioma, ele não possui instruções gramaticais ou explicações sobre nenhuma língua suportada, sendo seu uso recomendado como uma ferramenta *auxiliar* na qual o estudante pode expandir seu vocabulário e entendimento de uma língua por meio de consumo de conteúdos autênticos no idioma alvo, por conseguinte, seu público-alvo é o estudante em nível intermediário, ou maior, já com algum conhecimento básico da língua estudada.

3 METODOLOGIA

Este capítulo trata das metodologias utilizadas no desenvolvimento do trabalho. Inicialmente explicando sobre alguns design patterns utilizados no código, depois comentando um pouco sobre a ferramenta de construção de layouts para JavaFX, SceneBuilder. Por último, a seção 3.3 CrawlSpider discorre sobre a *CrawlSpider*, que é o tipo de aranha utilizada no desenvolvimento do DSPD.

3.1 Design Patterns

Design Patterns são padrões genéricos de software que visam resolver um problema recorrente em um projeto de software. Um padrão tem quatro elementos essenciais (Erich, et al, 1994): Um *nome*, o *problema*, a *solução* e a *consequência*. O nome é utilizado para referenciar ao padrão em questão como um todo. O problema é a explicação e o contexto do que se deseja resolver com o padrão. A solução é sobre os elementos de design, suas relações, responsabilidades e como se comunicam. Por último, um padrão contém consequências de ser aplicado.

Design patterns possuem três tipos principais (Erich, et al, 1994): comportamentais, de criação e estruturais. Comportamentais trabalham as formas como objetos interagem entre si e como suas responsabilidades são distribuídas. De criação diz respeito a todos os patterns que trabalham com criação de objetos. E estruturais lidam com composição de classes e objetos.

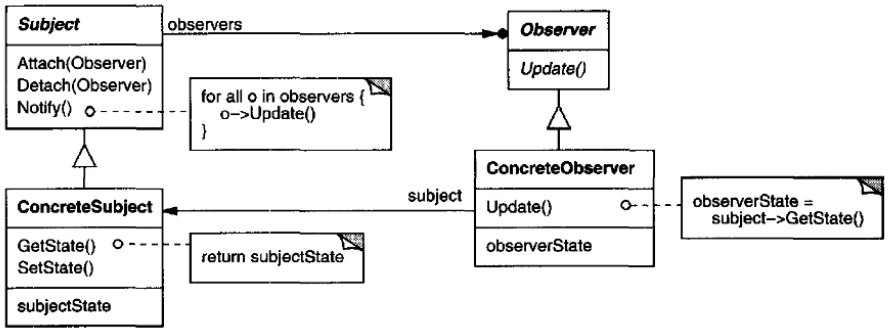
3.1.1 Observer

Este padrão, também conhecido como *publish-subscribe*, é um padrão comportamental em que há uma dependência muitos para um. Por conta do desacoplamento do sistema em múltiplos objetos independentes, acaba sendo necessário manter a consistência de algum modo.

O padrão *Observer* é um padrão em que várias classes podem se inscrever para receber notificações de mudança de uma classe observável. A visão geral do padrão pode ser observado na Figura 2 abaixo.

A Interface observável é chamada de *subject* ou *observable*, ela disponibiliza um mecanismo de adição e remoção de observáveis e conhece todos os objetos que estão inscritos para receber suas notificações. A interface *Observer* provém um método de update, que será excecutado quando uma alteração for notificada pelo *Subject*. Essas interfaces serão implementadas por classes concretas.

Figura 2 - arquitetura padrão observer



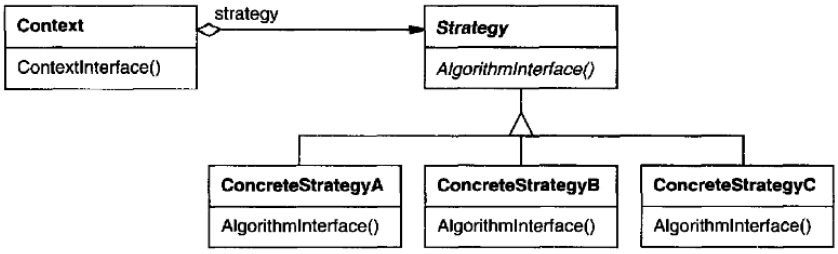
Fonte: Design Patterns: Elements of Reusable Object-Oriented Software (Erich, et al, 1994)

3.1.2 Strategy

Este padrão define uma família de algoritmos permutáveis, no qual o algoritmo que será executado é escolhido por alguma estratégia interna de negócio. É um padrão que pode ser usado quando múltiplas classes diferem apenas em comportamento, precisa-se de variantes diferentes de um algoritmo, ou quando uma classe tem muitos comportamentos possíveis para uma dada situação.

A Figura 3 traz a arquitetura do padrão estratégia. A interface *Strategy* contém uma interface comum para todas as estratégias concreta implementarem, o *Context* é a classe em questão que usa essa estratégia e será configurado de algum modo com a estratégia adequada.

Figura 3 - Arquitetura padrão Strategy

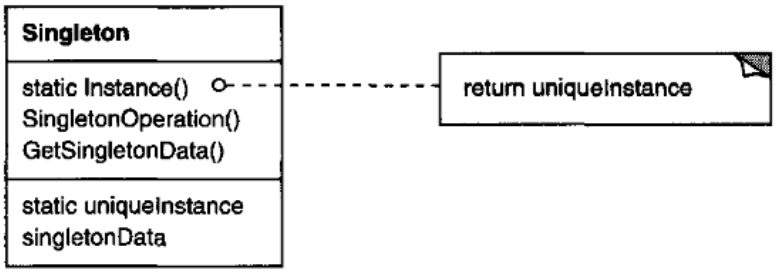


Fonte: Design Patterns: Elements of Reusable Object-Oriented Software (Erich, et al, 1994)

3.1.3 Singleton

Este padrão é utilizado para quando se necessita que uma classe possua apenas uma instância definida. Neste padrão, não é definido um inicializador público, mas sim um objeto estático da própria classe, sendo a instância única administrada internamente pela classe.

Figura 4 - Arquitetura padrão Singleton



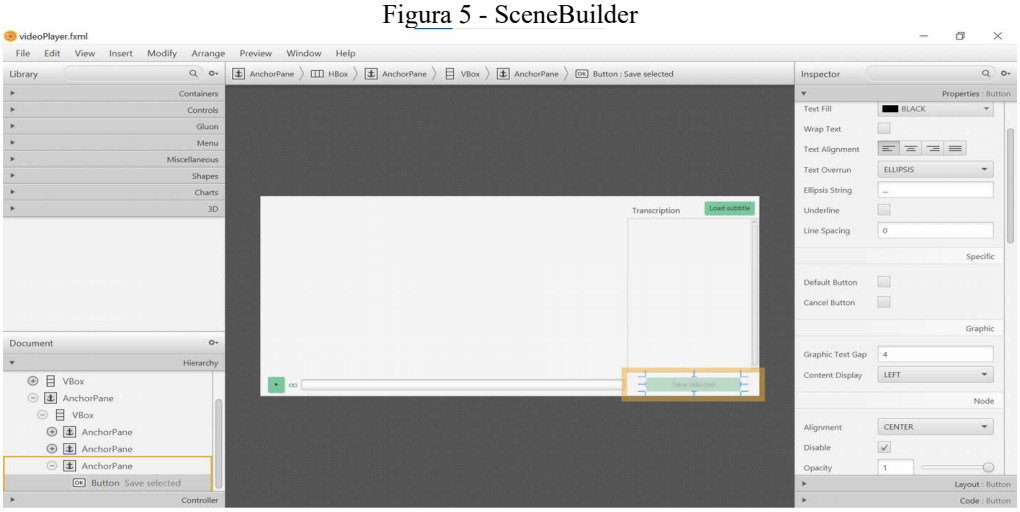
Fonte: Design Patterns: Elements of Reusable Object-Oriented Software (Erich, et al, 1994)

3.2 SceneBuilder

SceneBuilder é uma ferramenta para construções de layouts no JavaFX, que foi utilizada para a construção de interface gráfica do INTLIN. Usualmente, as interfaces do JavaFX são descritas em um arquivo estilo XML, o FXML. Visando agilizar o processo de construção de layouts, o SceneBuilder surge como uma ferramenta de *drag and drop*, permitindo rápida construção de telas sem necessitar especificar o FXML diretamente.

O ambiente de trabalho do SceneBuilder pode ser visualizado na Figura 5, no centro fica disposta a *viewport* com um *preview* do *layout*. Na esquerda, existe um *Accordion* de bibliotecas, onde é possível selecionar componentes para inserir na interface, também é possível encontrar a árvore de componentes, bem como um menu do controlador. Na direita estão as propriedades do componente selecionado, indo de estilos, textos alinhamentos à ações mais complexas de cliques ou outras interações de *input* sobre dito componente.

Figura 5 - SceneBuilder



Fonte: produção do autor

Sem a utilização do SceneBuilder, seria necessário especificar um FXML para montar a interface, o que requereria mais tempo, considerando a necessidade de escrever código manualmente.

3.3 CrawlSpider

CrawlSpider são um tipo de aranha oferecida pelo framework Scrapy, tipicamente usadas em *web crawling*, nesse tipo de aranha são definidos um conjunto de regras para extração de links e seguimento dos mesmos. Este tipo de aranha foi utilizado para extrair dados de dicionário do Wiktionary, para utilização no módulo de dicionário do INTLIN.

As regras levantadas para seguimento de links são regras para adentrar páginas de categorias e para entrar e extrair informações de páginas de definições. Tais regras são trabalhadas em mais detalhes em 6.1.3.2 Regras da aranha.

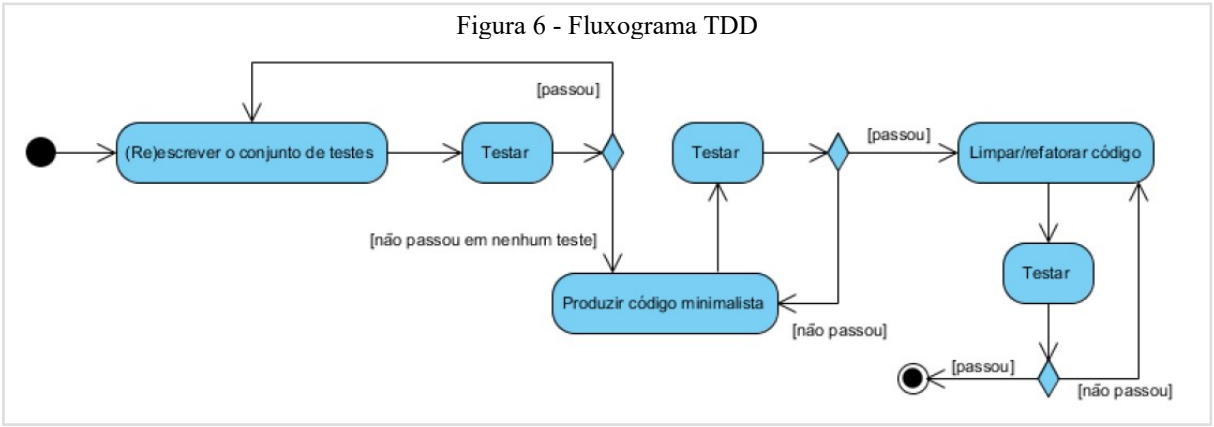
4 FUNDAMENTAÇÃO TEÓRICA

Este capítulo trata de conceitos teóricos básicos para o entendimento do trabalho. A 4.1 falará sobre TDD que foi uma abordagem utilizada para a construção de algumas partes do modelo do protótipo. Após isso será explicado sobre web scraping e o framework scrapy, conceitos fundamentais para entender o funcionamento do DSPD. Também serão tratados tópicos sobre repetição espaçada, que é um dos conceitos principais utilizados pelo INTLIN.

4.1 Test-Driven Development

Test-Driven Development é uma prática ágil proposta por Kent (Beck, 2002) em que o desenvolvimento dos testes unitários ocorre anteriormente a implementação do método a ser desenvolvido. A ideia principal é que este modo de desenvolvimento garantiria melhor qualidade, produtividade e um design mais limpo. O TDD segue o fluxograma demonstrado na figura 6.

Figura 6 - Fluxograma TDD



Fonte: Engenharia de Software – Conceitos e Práticas (Wazlawick, 2019)

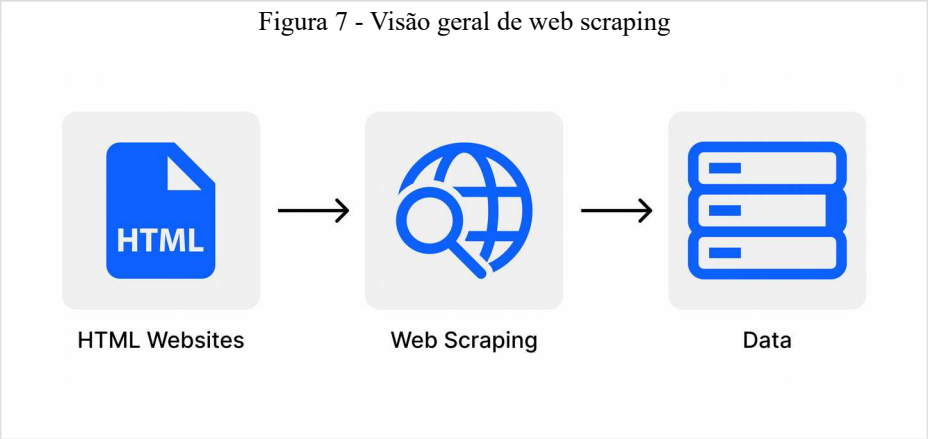
Primeiramente antes de desenvolver alguma funcionalidade, são escritos testes que deverão falhar, pois ela não estará implementada neste ponto. Na sequência são desenvolvidos códigos simples para passarem no teste, para por último refatorar ou polir o código, de modo a quando estes passarem no teste, pode-se dizer que a funcionalidade está correta.

4.2 Web scraping

Web scraping é uma forma de mineração de dados, que consiste em utilizar um rastreador, comumente chamado de “aranha”, para navegar páginas específicas da web coletando dados

presentes em páginas de modo não estruturado, e transformando-os em dados estruturados (Sirisuriya, 2015).

Figura 7 - Visão geral de web scraping



Fonte: Building a Python Web Scraping Project From Scratch[1]

Existem muitos softwares para se realizar *web scraping*, desenvolvidos em diversas linguagens para diversos sistemas operacionais. Para Python, por exemplo, o framework *Scrapy* é bastante poderoso e configurável, bem como é de fácil utilização.

4.2.1 Scrapy

Scrapy é um framework para web scraping e web crawling, escrito em Python multiplataforma para Linux, Windows e Mac. Ele é um framework bastante poderoso, com bastante configurações disponíveis, como *autothrottle*, *pipelines*, *proxies*, entre outros.

Esta seção trata de conceitos básicos do Scrapy utilizados neste trabalho. Primeiro, falar-se-á do conceito de itens, depois serão discutidos os seletores que podem ser usados para extrair dados do HTML da página. Também serão discutidos *pipelines*.

4.2.1.1 Item objects

Os dados extraídos de uma página web com o Scrapy são retornados como *itens*. O framework provê alguns tipos diferentes de itens (Items, Scrapy), como simplesmente um objeto dicionário comum, objetos *dataclasses*, objetos *attrs* e *item objects*, sendo este último o tipo de item tratado nesta seção.

1 Disponível em: <https://jovian.ai/alondes/python-web-scraping-project-guide>. Acesso em: 24/06/2022

O *item object* possui uma API similar a um dicionário Python, para acessar seus campos. Um exemplo, tirado da documentação do Scrapy (Items, Scrapy), para criar uma classe de item pode ser visualizado no trecho de código abaixo.

```
from scrapy.item import Item, Field

class CustomItem(Item):
    one_field = Field()
    another_field = Field()
```

Então, é possível usar este item como um objeto de dicionário, acessando seus valores da forma CustomItem[“one_field”], podendo escrever e ler deste modo.

4.2.1.2 Seletores

O Scrapy possui mecanismos para selecionar partes do HTML de páginas para poder extrair os dados desejados, sendo disponibilizados dois tipos de seletores pelo framework: o seletor CSS e o seletor XPath (Selectors, Scrapy).

O seletor CSS utiliza a linguagem de estilos CSS para selecionar partes do HTML que sejam contemplados pelo estilo especificado no seletor. Por exemplo, a linha de código `response.css("span.mw-headline")` seleciona qualquer tag que tenha como classe “mw-headline”.

O seletor XPath é utilizado para selecionar nodos em documentos XML, funcionando também com HTML. Para executar a mesma seleção do exemplo anterior com XPath, ter-se-ia `response.css("//span[@class=\'mw-headline\']")`.

4.2.1.3 Pipeline

Após um item ter sido extraído, ele é mandado para os pipelines, que são classes Python que implementam um método de processar um item, o *process_item*. Um pipeline pode ser usado para fazer qualquer tipo de tratamento com o dado após sua extração, usos comuns são descartar dados inválidos, limpar valores sujos vindos do HTML ou armazenar os itens em um banco de dados (Item Pipeline, Scrapy). É possível que hajam múltiplos pipelines em sequência, passando as alterações realizadas por um adiante para o próximo

O método principal que deve ser implementado em um pipeline é o *process_item*, este é responsável por fazer o tipo de tratamento desejado no item e mandá-lo a seguir para um próximo

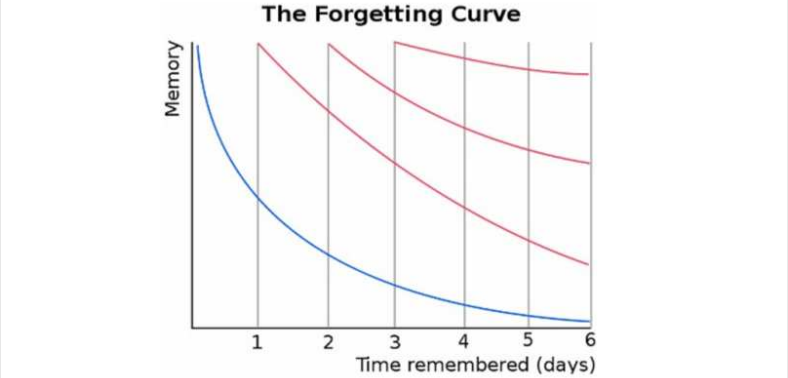
pipeline ou para um arquivo de saída. Também existem os métodos *open_spider* e *close_spider* que ocorrem, respectivamente, quando a aranha é aberta e fechada. Por último, o método *from_crawler* pode ser sobrescrito quando se deseja pegar informações vindas da aranha, e.g. atributos passados para a aranha por linha de comando

4.3 Repetição espaçada

Repetição espaçada é uma técnica de revisão de conteúdos, em que as revisões são espaçadas em períodos variantes de tempos, visando tomar proveito do *spacing effect*, fenômeno psicológico observado pela primeira vez por Hermann Ebbinghaus (1885)[2]. Repetição espaçada é costumeiramente realizada junto a flashcards, que são cartões com duas faces, uma contendo uma questão a se revisar e outra contendo uma solução.

Ebbinghaus também levantou o conceito de forgetting curve, que demonstra o quanto informações se perdem ao longo do tempo, e como a curva vai se achatando a cada repetição realizada. A Figura 8 ilustra a curva de esquecimento, bem como o processo de achatamento da mesma.

Figura 8 - forgetting curves



Fonte: How to improve your memory – Meta skills part 5[2]

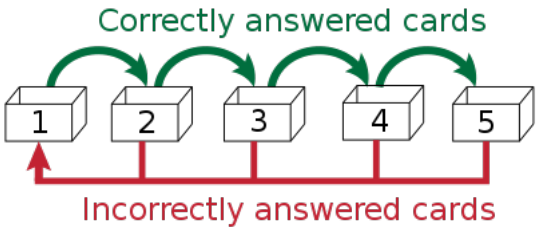
4.3.1 Sistema de Leitner

O algoritmo mais comum utilizado em repetição espaçada é o Sistema de Leitner, proposto por Sebastian Leitner na década de 70 em seu livro "*So lernt man lernen. Der Weg zum Erfolg*". O método consiste em utilizar 5 caixas que representam o quão bem o praticante se lembra de um conteúdo. Quando um conteúdo é colocado para revisão, ele entra na caixa de nível 1, e sempre que

2 Disponível em: <https://80000hours.org/2013/04/how-to-improve-your-memory/>. Acesso em: 25/06/2022

se realiza a revisão daquele conteúdo, em caso de conseguir relembrá-lo o cartão é colocado na caixa de nível acima. Quando uma revisão falha, o cartão retorna à primeira caixa.

Figura 9 - Leitner system



Fonte: Leitner system[3]

Originalmente, as caixas eram caixas reais e os cartões de papel ou papelão, bem como o agendamento para uma revisão se dava quando uma caixa ficava cheia, sendo as caixas de níveis maiores mais fundas para tornar a revisão mais espaçadas que de níveis menores. Presentemente, com computadores os cartões podem ser substituídos por flashcards armazenados em baralhos digitais, as caixas podem ser substituídas por níveis lógicos de proficiência, além do agendamento poder ser feito algoritmicamente para as revisões caírem para aproximadamente antes do estudante estar perto de esquecer o conteúdo, de acordo com a curva de esquecimento.

4.4 Questionário de usabilidade System Usability Scale

O *System Usability Scale*, também conhecido como SUS, é um modo de se mensurar o nível de usabilidade de um sistema, tendo sido desenvolvido por John Brooke (John, 1995). O questionário é constituído de 10 questões, sendo perguntas de índices ímpares aspectos positivos e de índices pares aspectos negativos. Cada questão é respondida em uma escala de 1 à 5, em que 1 é “discordo completamente” e 5 é “concordo completamente”.

O questionário pode gerar uma pontuação em um intervalo de 0 a 100, e é calculado da seguinte forma:

1. Para perguntas pares, subtrai-se a resposta do usuário por 5. (e.g. resposta = 3; contabiliza-se 2).
2. Para perguntas de índices ímpares, subtrai-se 1 da resposta do usuário. (e.g. resposta = 5; contabiliza-se 4).
3. Soma-se os valores das 10 perguntas
4. Multiplica-se por 2.5

3 Disponível em: <https://en.wikipedia.org/wiki/Leitner_system>. Acesso em: 25/06/2022

As questões que compõe o questionário podem ser averiguadas a seguir.

1. Eu penso que usarei este sistema com frequência
2. Acho o sistema desnecessariamente complexo
3. Penso que o sistema é fácil de usar
4. Acho que vou precisar de ajuda de um técnico para usar este sistema
5. Acho as funções deste sistema bem integradas
6. Encontro muitas inconsistências neste sistema
7. Imagino que as pessoas aprenderão rapidamente a usar este sistema
8. Acho o sistema imprático de usar
9. Senti-me confiante ao usar o sistema
10. Precisei aprender muitas coisas antes de ser capaz de operar o sistema

4.5 Aprendizagem de línguas

Dentro de métodos de aquisição de segunda língua, existem diversas perspectivas sobre que tipo de abordagem deve ser adotada, levanto em conta aspectos de como tal abordagem visualiza que uma pessoa aprende uma segunda língua (Patsy & Spada, 2006).

Esta seção trata da abordagem de behaviorismos, que trata da aprendizagem em termos de imitação, prática e reforço. O inatismo, que acredita que os seres humanos têm uma habilidade natural para desenvolver línguas. A perspectiva cognitiva, que são um conjunto de teorias e estudos que tentam entender como a mente humana desenvolve uma segunda língua. E por último, tratar-se-á da perspectiva sociocultural, a qual propõe que o desenvolvimento cognitivo se dá a partir de diversas condições socioculturais, incluindo o aprendizado de línguas. (Patsy & Spada, 2006)[4].

4.5.1 Perspectiva do behaviorismo

Esta abordagem de aprendizado de segunda língua, que foi muito forte durante as décadas de 40 à 70, toma como pilares a imitação, a prática e o reforço. Atividades de aprendizado dentro deste *framework* imitação e memorização de sentenças e padrões, visando os estudantes decorarem padrões de falas.

Aqui, aquisição linguística é vista como um desenvolvimento de hábitos, e deste modo, é pressuposto que padrões da língua materna do estudante interfeririam com os novos hábitos necessários para algum segundo idioma. Contudo, alguns estudos (Patsy & Spada, 2006) chegaram

4 Todas as subseções desta seção de “Aprendizado de línguas” serão baseados nessa referência (Patsy & Spada, 2006)

a conclusão de que erros cometidos por estudantes de alguma língua não eram previsíveis por comparação direta com o idioma materno do estudante.

Em torno da década de 70, esta perspectiva de criação de hábitos por imitação foi desfavorecida como explicação adequada para aprendizado de segunda língua.

4.5.2 Perspectiva do inatismo

A perspectiva do inatismo surge a partir de estudos de Noam Chomsky, em que todo ser humano teria a capacidade de desenvolver línguas com base em algum princípio nato universal. Tal princípio foi chamado por Chomsky de *Universal Grammar* (ou UG).

Os estudos de Chomsky visavam explicar o aprendizado de línguas para primeira língua, I.e. como uma criança desenvolvia sua língua. Contudo, outros pesquisadores partem desse conceito e o estendem para aquisição de segunda língua, Alguns argumentando que a UG se altera após aprender uma primeira língua, e outros que ela se manteria a mesma.

Um dos principais e mais influentes modelos que parte dessa teoria de Chomsky, seria o modelo proposto por Stephen Krashen. Neste modelo, é sugerido que pessoas adquirem línguas ao se exporem ao idioma com conteúdos que estejam no nível do estudante, ou levemente acima, é conhecido como “i+1”, onde i é o nível do estudante no idioma e o “+1” representa informações ainda não aprendidas levemente superior ao nível do estudante. O modelo de Krashen sofreu diversas críticas, conforme discutido em 1 INTRODUÇÃO, contudo suas ideias foram muito influentes e sua hipótese de *input compreensível* serviu como fonte de inspiração para metodologias de aprendizagem amplamente implementadas, como imersão e baseadas em conteúdos.

4.5.3 Perspectiva cognitiva

A perspectiva cognitiva sobre aprendizagem de línguas é composta por uma série de pesquisas e teorias da psicologia cognitiva que visam entender o desenvolvimento de segunda língua. Nesta perspectiva, aquisição de primeira e segunda língua são visto da mesma forma, como a realização de quatro grandes processos: percepção, memória, categorização e generalização. A diferença principal entre primeira língua e segunda que informações e conhecimentos anteriores podem moldar e influenciar a visão do estudante sobre a língua nova.

Existem diversas pesquisas na área, e múltiplas formas de entender o funcionamento do cérebro são descritas. Algumas fazem analogia do cérebro com um computador, realizando comparações da capacidade de aprender línguas às capacidades de computadores armazenar,

recuperar e processar dados. Outras pesquisas visam se apoiar em neurobiologia, tentando relacionar comportamentos na aprendizagem com atividade cerebral.

4.5.4 Perspectiva sociocultural

Nesta perspectiva, é defendido que o aprendizado de idiomas é regido por interações socioculturais e que a internalização da língua se daria por um indivíduo interagindo com um interlocutor dentro de sua "Zona proximal de desenvolvimento" (ZPD, *zone of proximal development*).

A ideia é similar ao "i+1" de Krashen, porém, enquanto o "i+1" tem foco em input, o ZPD foca na *interação* entre dois ou mais indivíduos, focando em um desenvolvimento mutuo de conhecimento para os envolvidos.

5 MÉTODOS E REQUISITOS

Esta seção tratará dos métodos e ferramentas usadas no desenvolvimento dos sistemas, bem como dos requisitos que se desejam implementar.

É importante esclarecer que este trabalho consiste de dois sistemas a parte, o INTLIN e o DSPD, já introduzidos na seção 1.2 OBJETIVOS. Os sistemas não se comunicam entre si, sendo apenas os dados extraídos pelo DSPD usados localmente pelo INTLIN. Para desenvolvê-los, métodos e requisitos diferentes foram adotados e serão detalhados a seguir.

5.1 Requisitos

Ambos INTLIN e DSPD possuem seus requisitos particulares, que serão tratados nesta seção. Os Requisitos do INTLIN dizem respeito a funcionalidades que serão prestigiadas pelo sistema, enquanto os do DSPD tratam de comportamento que a aranha deve seguir e alguns pontos de como será o modelo de dados extraídos.

5.1.1 Requisitos do INTLIN

Esta seção começa tratando os requisitos funcionais do sistema, separando eles pelos módulos contemplados, e então na sequência, serão discutidos alguns requisitos não-funcionais que o INTLIN deverá cumprir em seu desenvolvimento.

5.1.1.1 Requisitos funcionais

Como já discutido na proposta, o INTLIN possuirá três módulos principais, um de mídias, um de dicionários e um de SRS. Dentro do sistema haverá um certo nível de comunicação entre os módulos, de modo a automatizar o processo de estudo.

Para o módulo de mídias, deverá ser possível abrir mídias de áudio, vídeo e PDF, sendo as extensões possíveis de áudio e vídeo devem ser todas aquelas disponibilizadas pelo JavaFX, i.e. aif, aiff, wav e mp3 para áudio, e mp4, m4a, m4v, fxm e flv para vídeos. Este módulo também possuirá um submódulo de legendas, em que deve ser possível carregar arquivos de legenda com extensão .srt. O módulo de mídias deve permitir a carregamento simultâneo de arquivos de mídia do tipo: (i) Vídeo e legenda, (ii) Áudio e legenda, (iii) PDF e áudio, de modo a permitir vídeos com legendas, áudio com transcrição e audiobooks, respectivamente. Deve permitir, também, que sejam salvos trechos de logs das legendas no SRS, bem como salvar trechos de imagens de vídeos.

Para o módulo de dicionários, deve ser possível abrir um dicionário do formato de arquivo SQLite (.db) ou utilizar um parser para gerar o SQLite a partir de um arquivo JSON gerado pelo extrator. Deve ser possível pesquisar palavras no dicionário enquanto se usa o sistema e salvar entradas do mesmo em no SRS por meio de um clique de botão.

Para o módulo de SRS, deve ser possível que o usuário realize revisões agendada para o dia, e que ao realizá-la, um feedback com a resposta salva seja exibido, bem como uma nova data para revisar aquele cartão seja gerada. Nos cartões de revisões deve ser possível apresentar imagens e/ou texto. Deve ser possível criar múltiplos baralhos, a fim de permitir o usuário a separar as revisões por tópicos. Também deve ser permitido ao usuário criar *flashcards* manualmente, de modo que o estudante possa fazer anotações extras, não vistas em seus conteúdos, caso assim desejar.

Desta forma, levantou-se 7 requisitos associados ao módulo de mídias, 4 ao módulo de dicionários e 7 para o módulo de repetição espaçada. A Tabela 2 a seguir explicita tais requisitos.

Tabela 2 - Requisitos funcionais INTLIN

Módulo de mídias	
RF01	O sistema deve carregar arquivos de vídeo
RF02	O sistema deve carregar arquivo de áudio
RF03	O sistema deve carregar arquivo PDF
RF04	Com áudio ou vídeo carregados, o sistema deve carregar legendas (.srt)
RF05	Com PDF carregado, o sistema deve carregar arquivos de áudio
RF06	Com vídeo carregado, o sistema deve salvar capturas de tela para adicionar em baralhos do SRS
RF07	Com legendas carregadas, o sistema deve permitir que sejam selecionadas uma ou múltiplas entradas no log de legendas para salvar em um baralho do SRS
Módulo de dicionário	
RF08	O sistema deve carregar um dicionário a escolha do usuário
RF09	O sistema deve conseguir realizar parsing de arquivos JSON gerados na saída do DSPD
RF10	O sistema deve exibir resultado de consultas de palavras pesquisadas pelo usuário
RF11	O sistema deve salvar uma entrada de dicionário selecionada, para adição em baralhos do SRS
Módulo de repetição espaçada	
RF12	Deve ser possível realizar revisões de cartões salvos agendados para o dia
RF13	Após uma revisão, o sistema deve dar feedback de quando o cartão estará disponível para revisão novamente
RF14	Após uma revisão, o sistema deve dar feedback com a resposta salva no cartão
RF15	O sistema deve conseguir exibir textos nos cartões de revisão
RF16	O sistema deve conseguir exibir imagens no cartão de revisão

RF17	O sistema deve permitir a criação de baralhos
RF18	O sistema deve permitir a criação de cartões manualmente

5.1.1.2 Requisitos não-funcionais

A Tabela 3 expõe os requisitos não funcionais do INTLIN. Foram levantados quatro requisitos dizendo respeito a implementação, sobre a tecnologias e padrão a serem utilizados. Um requisito sobre o produto em si, que deve ser completamente utilizável sem necessitar de internet. E o último tratando de um aspecto ético que não deve ser utilizado nenhum dado protegido sobre direitos autorais, isto é, no modulo de dicionários só devem ser utilizados dados que foram adquiridos de fontes que permitem o uso de seus dados.

Tabela 3 - Requisitos não-funcionais INTLIN

RNF01	O sistema deve ser implementado em Java
RNF02	Deve ser utilizada programação orientada a objetos no sistema
RNF03	Deve ser utilizado o padrão de projeto MVC
RNF04	Deve ser utilizado um banco SQLite para os módulos de dicionário e SRS
RNF05	O sistema deve ser completamente offline
RNF06	O sistema deve apenas utilizar dados de direitos autorais livres

5.1.2 Requisitos do DSPD

O DSPD, em questão de funcionalidade, é um sistema bem mais simples, tendo poucos requisitos funcionais. Esta seção tratará dos requisitos para essas poucas funcionalidades, e na sequência cobrirá os requisitos não-funcionais.

5.1.2.1 Requisitos funcionais

O DSPD faz o que se espera de um extrator, ele manda requisições a um servidor e faz parse do HTML para extrair os dados desejados de modo estruturado. Além disso, ele tem como funcionalidade limpar eventuais sujeiras nos dados e inseri-los em banco de dados, bem como deve ser possível que se escolha o idioma que se deseja extrair definições.

Deste modo, elencou-se 6 requisitos funcionais, listados na Tabela 4.

Tabela 4 - Requisitos funcionais DSPD

RF01	O extrator deve fazer requisições ao Wiktionary
RF02	O extrator deve extrair dados relevantes da página requisitada ao Wiktionary
RF03	O extrator deve permitir que se escolha qual idioma se extrairá definições
RF05	O extrator deve verificar e limpar um item após sua extração
RF06	O extrator deve inserir um item extraído em um banco de dados

5.1.2.2 Requisitos não-funcionais

O DSPD deverá ser um extrator respeitoso que navega lentamente pelas páginas do site, de modo a evitar um sobrecarregamento do servidor, além de que o arquivo robots.txt do Wiktionary deve ser respeitado.

Os modelo de dados para extração devem ser genéricos de modo a poder se extrair dados de múltiplos idiomas. Os dados extraídos serão armazenados em bancos de dados SQLite, para uso local no INTLIN, bem como será gerado um arquivo JSON dos dicionários.

Para geração de tais dados será utilizada a funcionalidade de pipelines do Scrapy. Dois pipelines devem ser utilizados, um para limpar dados e outros para inserção de itens no SQLite.

Assim, os requisitos não funcionais estão evidenciados na Tabela 5, a seguir.

Tabela 5 - Requisitos não-funcionais DSPD

RNF01	O extrator deve fazer requisições lentas ao servidor
RNF02	O extrator deve obedecer as especificações do arquivo robots.txt
RNF03	O Modelo de dados extraído deve ser genérico
RNF04	O Banco de dados a ser utilizado deve ser o SQLite
RNF05	O tipo de arquivo de saída do extrator deve ser JSON
RNF06	Deve se utilizar o mecanismo de pipelines do Scrapy para limpeza dos dados
RNF07	Deve se utilizar o mecanismo de pipelines do Scrapy para inserção dos dados no SQLite

5.2 MÉTODOS

Esta seção destaca as principais tecnologias e formas de desenvolvimento utilizadas na produção do trabalho.

5.2.1 Métodos para o INTLIN

O aplicativo será desenvolvido utilizando a linguagem de programação Java acompanhado da plataforma JavaFX, dentro do ambiente de desenvolvimento integrado Netbeans. A interface gráfica com o JavaFX será modelada com a ferramenta SceneBuilder. A escolha destas tecnologias

se dá pelo fato do JavaFX ser uma ferramenta para frontend feito em JAVA mais robustas que outras disponíveis, como Java Swing. O SceneBuilder é uma ferramenta de *drag and drop* bem completa, sua utilização renderá um ganho em produtividade, não sendo tão necessário trabalhar com os arquivos estilo XML do JavaFX.

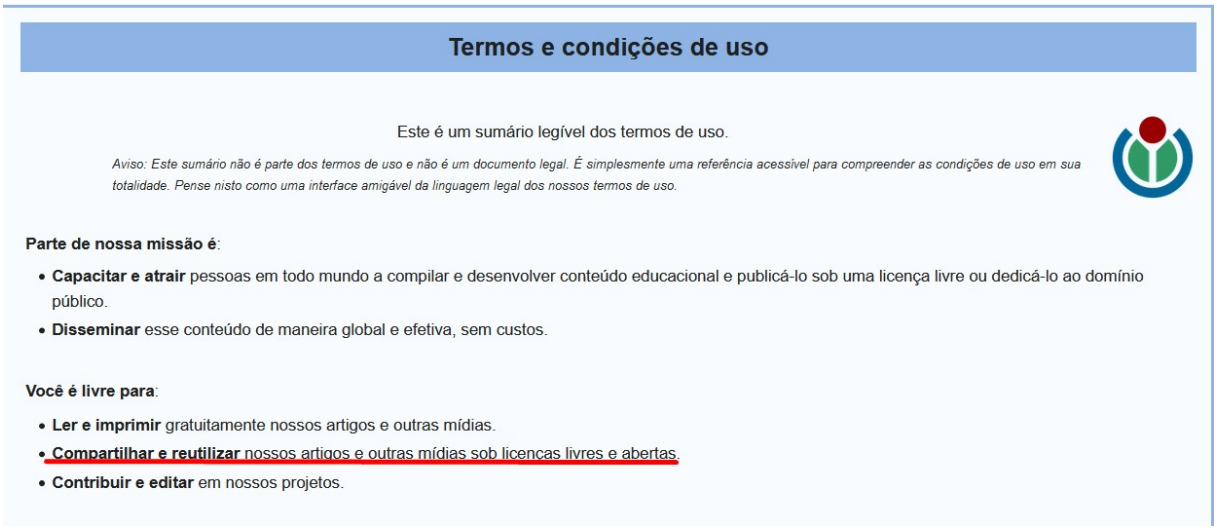
Também vale destacar que para classes de modelo do sistema será utilizada a técnica de desenvolvimento de software *Test-Driven Development*. Esta metodologia será usada no modelo visando uma melhor qualidade e corretude das funcionalidades desenvolvidas, tentando cobrir a maior quantidade possível de comportamento esperado pelo sistema.

5.2.2 Métodos para o DSPD

O DSPD será desenvolvido em Python usando o framework *Scrapy*, devido a sua robustez e facilidade de utilização. Serão utilizados várias funcionalidades do framework, como definições de itens de extração, pipelines para trabalhar com dados e configurações diversas de extração.

O web scraper seguirá os preceitos éticos de não desobedecer o arquivo robots.txt, fará requisições lentas para o site, evitando sobrecarregar o servidor. Importante ressaltar que nenhum direito autoral será ferido no processo, a Figura 10 possui um resumo dos termos de uso da Wikimedia Foundation, no qual declara que é permitido o reúso e compartilhamento de seus dados sob licença livre.

Figura 10 - Direitos autorais da wiki



Fonte: termos e condições da Wikimedia Foundation[5]

5 Disponível em: <https://foundation.wikimedia.org/wiki/Terms_of_Use/pt-br> Acesso em: 14/08/2021

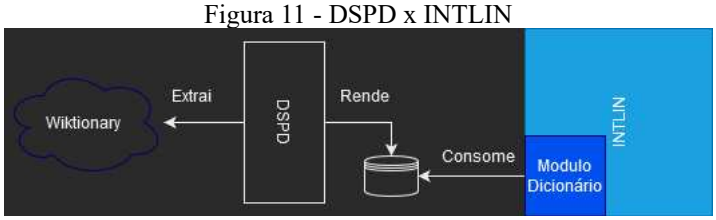
6 DESENVOLVIMENTO

Esse capítulo detalha o desenvolvimento de todo o projeto. Primeiramente, entrando em detalhes sobre a aranha, seus comportamentos, dados a serem extraídos e seus detalhes de implementação. Na sequência, detalhar-se-á o desenvolvimento do INTLIN, tratando sobre cada módulo individualmente, bem como suas características e particularidades de desenvolvimento.

É importante ressaltar que o DSPD não é uma parte do INTLIN, ele é um sistema que rodará previamente para coletar dados que, posteriormente, serão utilizados pelo outro sistema. Assim, ele não funcionará como uma espécie de backend para o INTLIN, ele é um sistema desacoplado que também poderia ser utilizado para outros fins além da coleta de dados para o INTLIN.

Para efeitos deste trabalho, o DSPD foi utilizado para gerar dados para o modulo de dicionário do INTLIN. Ele gera vários dados, que são adicionados em um banco local SQLite, que pode ser posteriormente transferido para o INTLIN, e então usado, de mesmo modo, localmente pelo protótipo. A Figura 11 ilustra o processo.

Figura 11 - DSPD x INTLIN



Fonte: produção do autor

6.1 DICTONARY SPIDER (DSPD)

O primeiro passo, antes de construir o INTLIN, é desenvolver um extrator de dados para montar um dicionário a ser usado pelo sistema. Esta seção tratará do desenvolvimento do DSPD, iniciando com modelagem e definições e seguindo para especificações da implementação. Esta seção contemplará aspectos de comportamentos do DSPD, dados extraídos e detalhes da sua implementação.

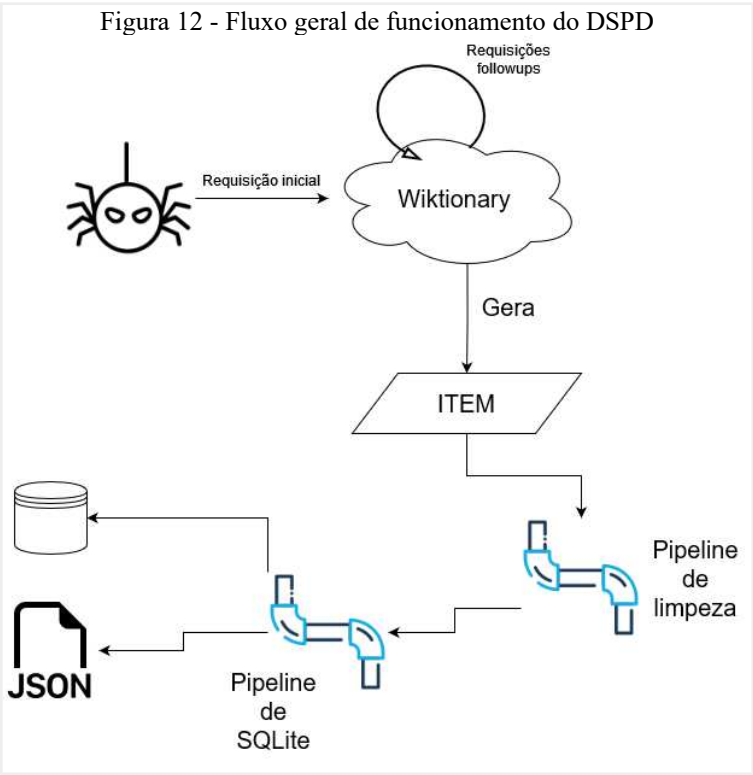
6.1.1 Comportamento da aranha

O DSPD, *Dictonary Spider*, é um extrator desenvolvido em Python com o uso do framework Scrapy. Seu fluxo de funcionamento básico, explicitado pela Figura 12, começa com a aranha requisitando, ao Wiktionary, endereços iniciais de algum idioma específico. A partir dos

links iniciais, a aranha segue para múltiplas páginas, em que realizará o *parsing* dos itens, e então os enviará para os dois *Pipelines*. O primeiro *pipeline* trata da remoção de dados inválidos e sujeiras que ocorrem em algumas páginas do Wiktionary, o segundo tem a função de inserir os itens coletados em um SQLite, bem como adicionar o item a um JSON de saída da execução.

Para evitar sobrecarregamento do servidor do Wiktionary, a aranha é bem conservadora, fazendo requisições lentas ao servidor, não requisitando nada em menos de 4 segundos, também utiliza um *middleware* de *autothrottle*, que regula a velocidade de requisição de acordo com o tempo de resposta do servidor (AutoThrottle, Scrapy). Acima disso, ela também obedece o arquivo robots.txt, a fim de ser respeitosa com o site e evitar banimento, visto que tal arquivo explicita que robôs não devem adentrar em certas páginas.

Figura 12 - Fluxo geral de funcionamento do DSPD



Fonte: produção do autor

6.1.2 Dados extraídos

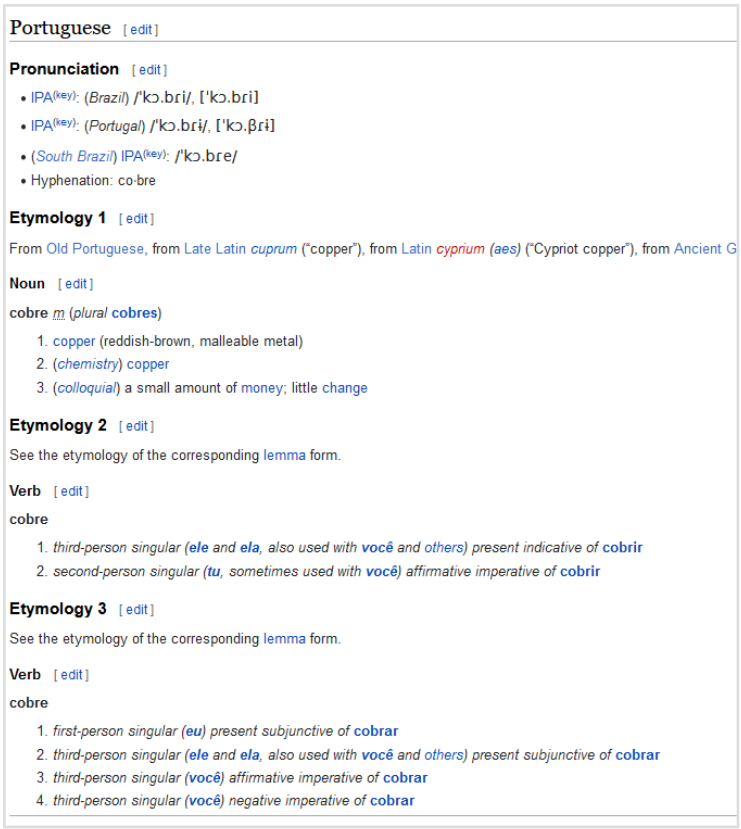
As definições de palavras foram extraídas de páginas do Wiktionary e colocadas em um *item* que tem a forma de um elemento de arquivo JSON, assim posteriormente, um JSON poderá ser gerado com todos os *itens*. Também é realizado um *parsing* sobre tais itens para inseri-los em um SQLite.

6.1.2.1 Definição dos dados

Primeiro, realizou-se uma análise em que tipos de informações estão presentes em uma página do Wiktionary, bem como definir o que se deve extrair. A Figura 13, a seguir, possui um exemplo de uma página do Wiktionary e, pode-se observar que para uma mesma string, é possível que ocorram múltiplas *entradas* no dicionário, delimitado pelas classes gramaticais. São, de fato, entradas diferentes, o cobre substantivo não deve ser considerado a mesma entrada que o verbo cobrir ou cobrar.

Similarmente, dentro de uma *entrada* é possível que ocorram múltiplas *definições*, vide como cobre substantivo possui três definições.

Figura 13 - Exemplo de página do Wiktionary com múltiplas entradas



Portuguese [edit]

Pronunciation [edit]

- IPA(key): (*Brazil*) /ˈkɔ.bɾi/, [ˈkɔ.bɾi]
- IPA(key): (*Portugal*) /ˈkɔ.bɾi/, [ˈkɔ.βɾi]
- (*South Brazil*) IPA(key): /ˈkɔ.bɾe/
- Hyphenation: co·bre

Etymology 1 [edit]

From Old Portuguese, from Late Latin *cuprum* (“copper”), from Latin *cyprium (aes)* (“Cypriot copper”), from Ancient G

Noun [edit]

cobre *m* (*plural* **cobres**)

1. copper (reddish-brown, malleable metal)
2. (*chemistry*) copper
3. (*colloquial*) a small amount of money; little change

Etymology 2 [edit]

See the etymology of the corresponding lemma form.

Verb [edit]

cobre

1. *third-person singular (**ele** and **ela**, also used with **você** and others) present indicative of* **cobrir**
2. *second-person singular (**tu**, sometimes used with **você**) affirmative imperative of* **cobrir**

Etymology 3 [edit]

See the etymology of the corresponding lemma form.

Verb [edit]

cobre

1. *first-person singular (**eu**) present subjunctive of* **cobrar**
2. *third-person singular (**ele** and **ela**, also used with **você** and others) present subjunctive of* **cobrar**
3. *third-person singular (**você**) affirmative imperative of* **cobrar**
4. *third-person singular (**você**) negative imperative of* **cobrar**

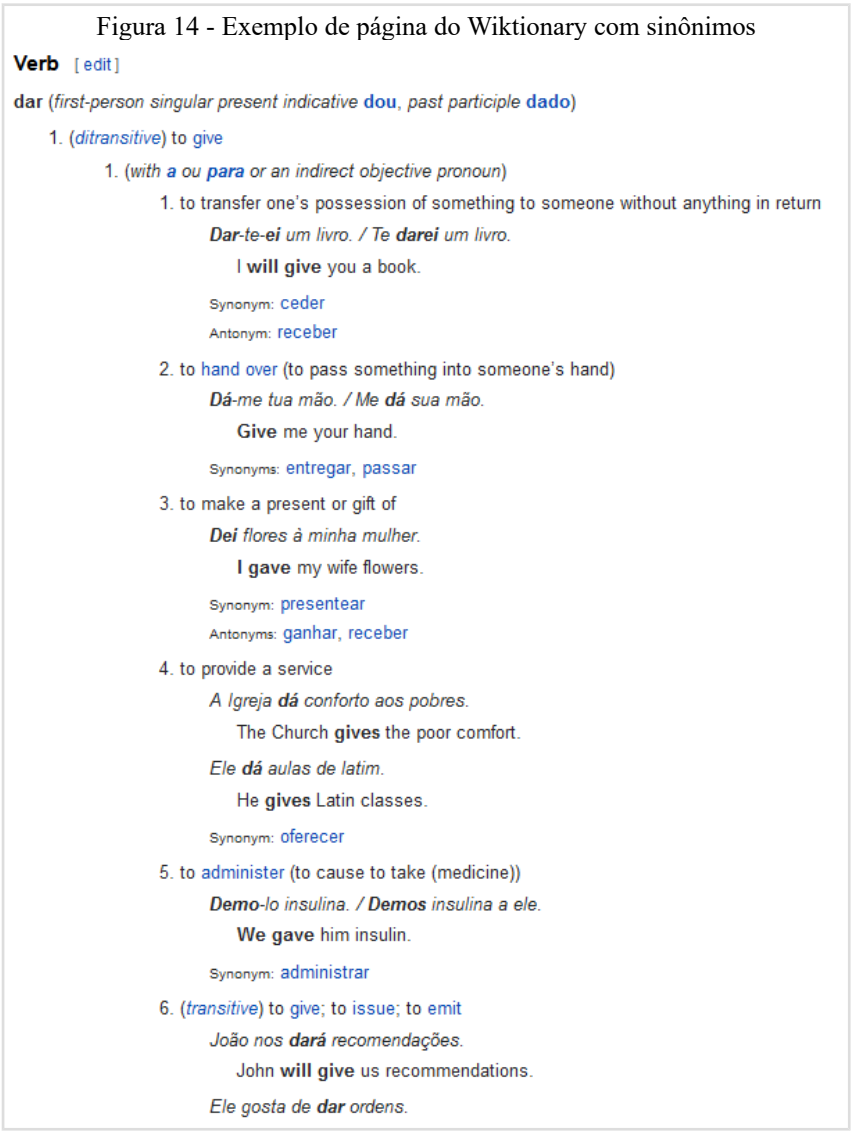
Fonte: Tela capturada pelo autor[6]

Outrossim, *definições* específicas de uma entrada podem possuir *atributos* próprios, tais como sinônimos e antônimos, a Figura 14 evidencia esse caso para o verbo dar, do português. E, de fato, essas informações pertencem as *definições* e não a *entrada* do verbo dar. Como exemplo, o sinônimo da definição 5 não se encaixa na definição número 1, e vice-versa. Adicionalmente, na

6 Disponível em: <https://en.Wiktionary.org/wiki/cobre#Portuguese> Acesso em: 07/03/2022

mesma figura, próximo aos sinônimos e antônimos, é possível perceber alguns textos sem marcadores, nesta figura são somente exemplos, contudo ocasionalmente aparecem algumas outras informações não tão relevantes. Assim, para esse extrator, optou-se por extrair qualquer informação de uma definição, fora sinônimos e antônimos, como sendo "extras" daquela definição.

Figura 14 - Exemplo de página do Wiktionary com sinônimos



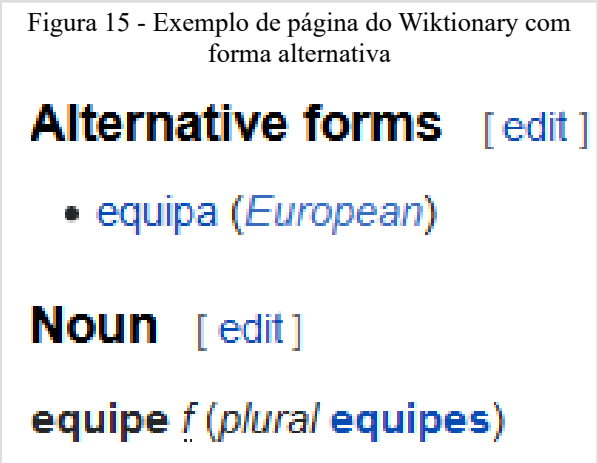
Fonte: Tela capturada pelo autor[7]

Por último, algumas páginas apresentam uma forma alternativa. Analisando a Figura 15, tem-se a forma alternativa de equipe como sendo equipa. Na página do Wiktionary, essa forma alternativa não aparece agrupada com classe gramatical, sendo declarada como forma alternativa global para todas as entradas de string "equipe". Isso, na verdade, é uma imperfeição no modelo do Wiktionary, visto que, nessa mesma página é possível encontrar equipe como o verbo *equipar* conjugado, o que evidentemente não deveria ter a forma alternativa "equipa". Infelizmente, não há

7 Disponível em: <https://en.Wiktionary.org/wiki/dar#Portuguese> Acesso em: 07/03/2022

informações na página que indiquem para a aranha à que definição a forma alternativa deveria ser inclusa. Assim, optou-se por respeitar o modelo do Wiktionary e adicionar a forma alternativa para cada string que aparecem sob essa forma alternativa.

Figura 15 - Exemplo de página do Wiktionary com forma alternativa



Fonte: Tela capturada pelo autor[8]

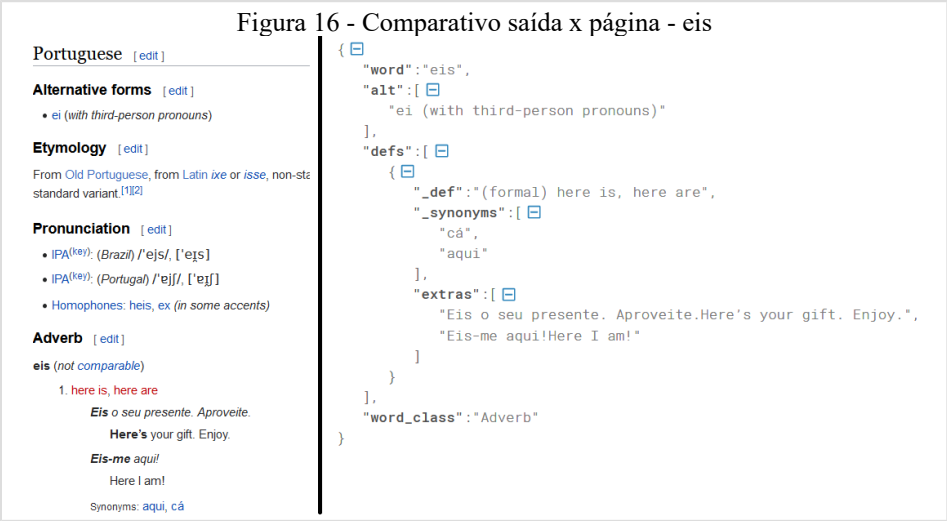
Desta forma, ao entrar em uma página os dados a serem extraídos são: (i) A palavra. (ii) Forma alternativas. (iii) Classe gramatical. (iv) Definições. (v) Sinônimos, Antônimos e Extras. (vi) Não mencionado, mas também é extraído o gênero da palavra, quando houver. As demais informações presentes nas páginas, como etimologia e pronuncia, foram julgadas desnecessárias por hora e não serão extraídas.

6.1.2.2 Formato dos dados

Com os dados a serem extraídos definidos, é possível mostrar como o JSON dos dados se comporta. As Figuras 16 e 17 trazem um comparativo *side-by-side* de páginas do Wiktionary e itens do JSON de saída gerado pela aranha.

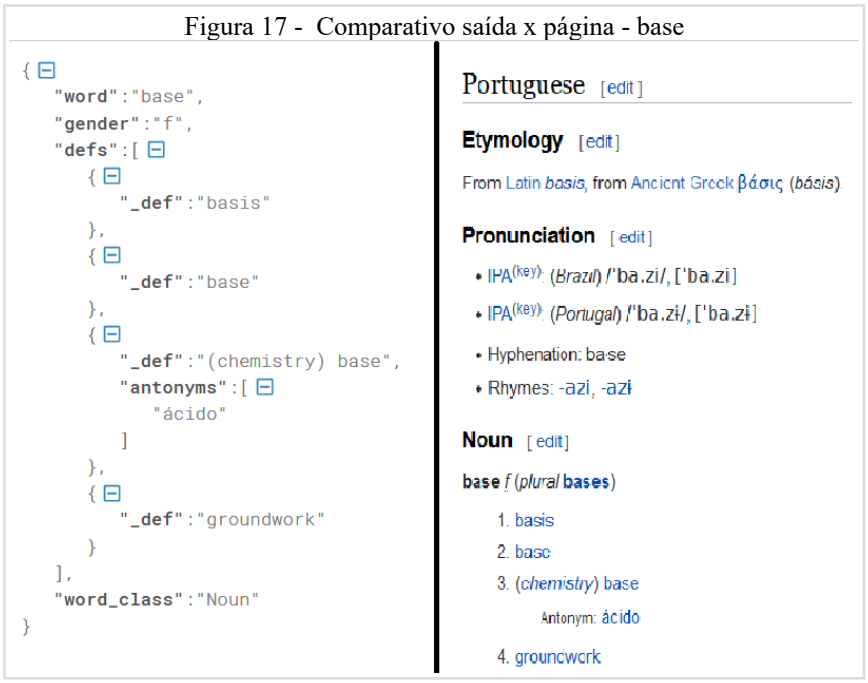
8 Disponível em: <https://en.Wiktionary.org/wiki/equipe#Portuguese> Acesso em: 07/03/2022

Figura 16 - Comparativo saída x página - eis



Fonte: Tela capturada pelo autor[9]

Figura 17 - Comparativo saída x página - base



Fonte: Tela capturada pelo autor[10]

Desta forma, pode-se chegar ao seguinte esquema para os arquivos JSON de saída. haveriam campos para palavra (String, obrigatório), gênero (String, opcional), formas alternativas (Lista de strings, opcional), classe gramatical (String, obrigatório), definições (Lista de JSON, obrigatório). Cada definição, por sua vez, tem como valor um JSON próprio, contendo campos para uma definição (String, obrigatório), sinônimos (String, opcional), antônimos (String, opcional) e extras (String, opcional). O trecho de código a seguir contempla o JSON, sendo o símbolo de interrogação no tipo de dados um denotador de optativo.

9 Disponível em: <https://en.Wiktionary.org/wiki/eis#Portuguese> Acesso em: 07/03/2022

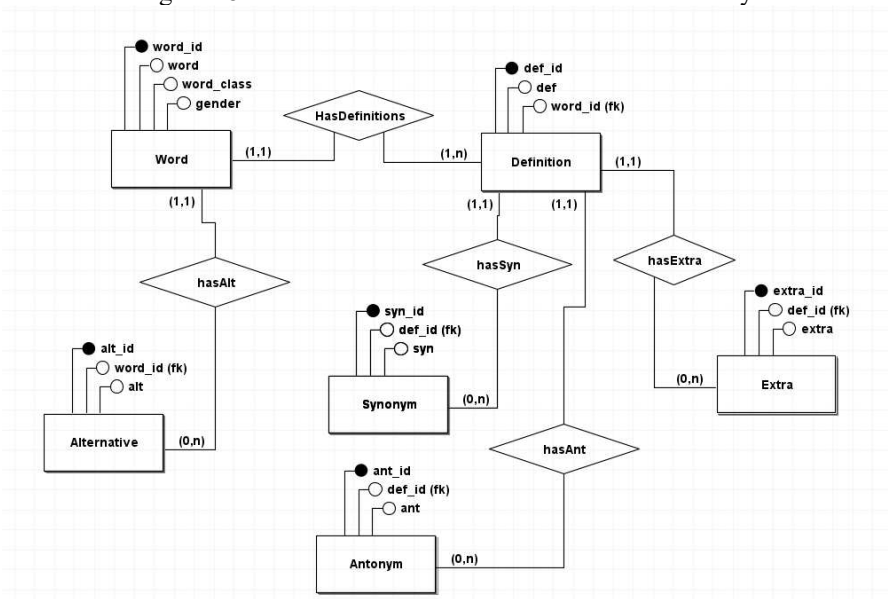
10 Disponível em: <https://en.Wiktionary.org/wiki/base#Portuguese> Acesso em: 07/03/2022

```
{
    word: String,
    gender: String?,
    alt: Lista<String>?,
    defs: Lista<{
            _def: String,
            antonyms: Lista<String>?,
            synonyms: Lista<String>?,
            extras: Lista<String>?,
        }>
    word_class: String,
}
```

Este modelo de dados é bastante genérico e pode comportar definições de diversos idiomas. Alguns campos, como *gender,* podem ser vazios, o que permite que se extraia tanto definições de idiomas com gênero, como sem gênero. Contudo, ele não é adequado para todos os idiomas, pois alguns podem necessitar de alguns campos não inclusos, o que levaria a uma necessidade de estender esse modelo, por exemplo um idioma como mandariam poderia querer um campo para pinyin (leitura da palavra).

Além desta saída em JSON, os dados também são inseridos em um SQLite, que teve seu modelo definido a partir do JSON. Como já mencionado o bot possui um pipeline que insere os itens extraídos em um banco de dados. Um SQLite torna mais eficiente para a aplicação realizar consultas, comparativamente a consultar diretamente do JSON.

Figura 18 - Modelo base de dados extraído do Wiktionary



Fonte: Produção do autor

6.1.3 Desenvolvimento do DSPD

Durante o desenvolvimento, ocorreram algumas mudanças na aranha, resultando em três versões distintas. A primeira delas era uma versão mais estática em que as requisições eram definidas manualmente em código. A segunda versão automatizou a coleta de links, de modo a fazer a aranha requisitar páginas a partir de qualquer link encontrado, contando que esses obedecessem um conjunto de regras. A terceira versão adiciona uma nova regra que previne a coleta de um tipo de dado que "poluía" os resultados finais.

É impossível garantir que não há erros nos dados extraídos, devido a enorme gama de páginas contida no Wiktionary, é muito difícil garantir que alguma delas não possua o HTML diferente, levando a algum erro na extração. Contudo, levando em consideração a grande quantidade de refatorações realizadas, pode-se alcançar um bot que gera bastante dados de qualidade, não tendo sido encontrado nenhuma saída absurda na última versão da aranha.

6.1.3.1 Configurações da aranha

Foram definidos uma série de valores a serem usado pela aranha dependente a qual idioma está se extraindo os dados, tais valores podem ser visualizados na Figura 19. Cada língua é identificada unicamente pelo código de 2 dígitos do padrão ISO 639-1[11].

O atributo *language* é usado para encontrar o header que inicia as definições do idioma desejado dentro do Wiktionary. O atributo de *word_classes* serve para a aranha encontrar as entradas específicas dentro da região do idioma, esse atributo existe pois não há diferença entre os cabeçalho de definições e outros cabeçalhos, e não se deseja, por exemplo, coletar a etimologia de uma palavra como sendo uma definição de classe gramatical "Etymology". Por isso as classes gramáticas são definidas manualmente. Essa lista de possíveis classes não são um valor global, mas de fato um atributo interno do idioma, isso se dá pelo fato de alguns idiomas poderem ter classes gramaticais diferentes, por exemplo, uma língua como japonês não possui artigos e possuiria uma classe de "partículas" não existente em muitas línguas.

O último atributo é o *start-url-crawler*, esse é o ponto de partida da aranha. A escolha foi por uma página geral de categoria dos idiomas, pois a escolha de seguir links se dá de forma automática pela aranha, sendo necessário apenas um ponto de partida que leve a várias páginas do idioma.

[11] Disponível em: <https://www.iso.org/iso-639-language-codes.html>. Acesso em: 08/03/2022

Figura 19 - Valores de configuração aranha

```
languages = {
    'pt': {'language': "Portuguese",
           'word_classes': ["Phrase", "Noun", "Verb", "Adjective", "Pronoun", "Article",
               "Contraction", "Preposition", "Proverb", "Proper noun",
               "Adverb", "Conjunction", "Numeral", "Interjection", "Symbol", "Suffix", "Prefix"],
           'start-url-crawler': ['https://en.wiktionary.org/wiki/Category:Portuguese_language'],
       },
    'es': {'language': "Spanish",
           'word_classes': ["Phrase", "Noun", "Verb", "Adjective", "Pronoun", "Article",
               "Contraction", "Preposition", "Proverb", "Proper noun",
               "Adverb", "Conjunction", "Numeral", "Interjection", "Symbol", "Suffix", "Prefix"],
           'start-url-crawler': ['https://en.wiktionary.org/wiki/Category:Spanish_language'],
       },
    'fr': {'language': "French",
           'word_classes': ["Phrase", "Noun", "Verb", "Adjective", "Pronoun", "Article",
               "Contraction", "Preposition", "Proverb", "Proper noun",
               "Adverb", "Conjunction", "Numeral", "Interjection", "Symbol", "Suffix", "Prefix"],
           'start-url-crawler': ['https://en.wiktionary.org/wiki/Category:French_language'],
       },
```

Fonte: Produção do autor

Além destas configurações definidas manualmente, o Scrapy traz uma série de configurações em um arquivo de *settings.py*, de modo geral as configurações selecionadas nesse arquivo, para este projeto, foram opções que tornam a aranha mais conservadora, visando não sobrecarregar o servidor do Wiktionary. Foram configuradas as opções de fazer a aranha obedecer o arquivo *robots.txt*, o qual impede a aranha de entrar em páginas que não são permitidas para bots. Foi configurado o tempo mínimo de delay entre requisições como 4 segundos, o que deixa a aranha bem lenta, mas garante que não gerará distúrbios no servidor. Também foi habilitado o mecanismo de autothrottle, que é uma funcionalidade que serve para regular a velocidade de requisições de acordo com o tempo de resposta do servidor, i.e., caso o Wiktionary demore para responder uma requisição, a aranha passará a mandar requisições mais lentas, até o tempo de resposta do servidor estabilizar. A Figura 20, a seguir, demonstra as configurações do autothrottle.

Figura 20 - Configurações de autothrottle

```
# Enable and configure the AutoThrottle extensio
# See https://docs.scrapy.org/en/latest/topics/a
AUTOTHROTTLE_ENABLED = True
# The initial download delay
AUTOTHROTTLE_START_DELAY = 10
# The maximum download delay to be set in case o
AUTOTHROTTLE_MAX_DELAY = 120
# The average number of requests Scrapy should b
# each remote server
AUTOTHROTTLE_TARGET_CONCURRENCY = 0.5
# Enable showing throttling stats for every resp
AUTOTHROTTLE_DEBUG = True
```

Fonte: Produção do autor

Também é declarada uma lista de domínios permitidos, sendo aqui apenas o Wiktionary (en.wiktionary.org/) permitido, pois não se deseja que a aranha saia internet afora em direção outros sites.

6.1.3.2 Regras da aranha

O framework do Scrapy disponibiliza diversos tipos de aranhas, uma delas é a *CrawlSpider*, neste modo de aranha, são definidos um conjunto de regras para extração de links e seguimento dos mesmos (Spiders, Scrapy), este é o tipo de aranha usada atualmente pelo DSPD. Nos primeiros esboços do DSPD era utilizada a aranha padrão, no qual o seguimento dos links eram definidos via código, contudo, ela se mostrou ineficiente, não gerando uma quantidade boa dados. Então, optou-se pela *CrawlSpider*, com ela, necessita-se apenas de um link inicial qualquer e um conjunto de regras bem definidas para a que a aranha adentre em todas as páginas que encontrar de um idioma especificado. Também vale ressaltar que um link uma vez visitado, não será acessado novamente.

As regras definidas podem ser vistas na Figura 21. É possível observar na Figura abaixo duas regras sendo definidas, ambas recebem um extrator de links, esse que é responsável por encontrar links que deverão ser seguidos pelo bot. Tais extratores estão usando seletores css, para selecionar partes específicas da página, ou seja, eles coletarão os links em qualquer tag capturada pelos seletores especificados.

Figura 21 - Regras de seguimento de links do DSPD

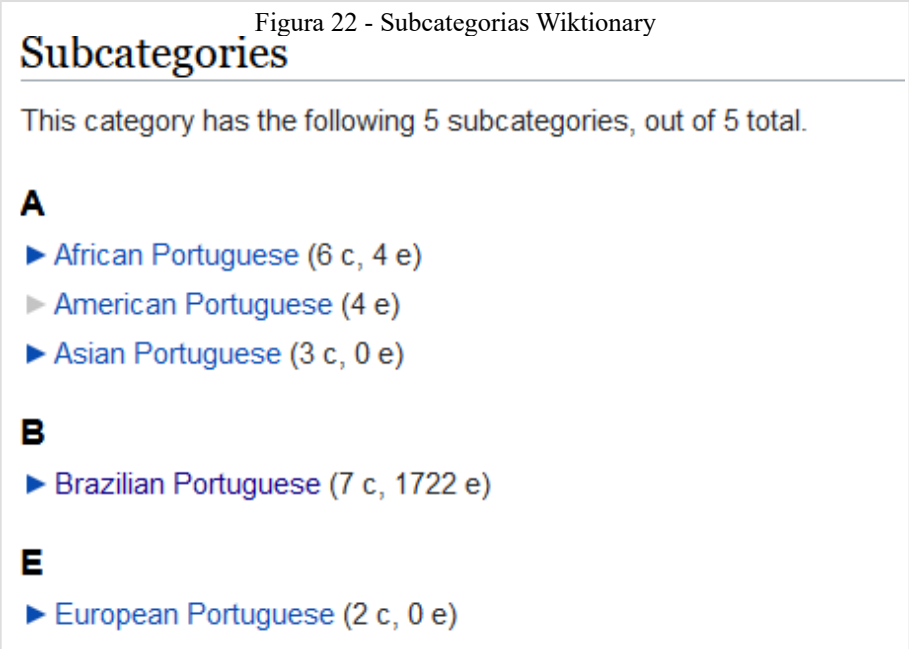
```
super(dictSpider, self).__init__(*a, **kw)
self.start_urls = languages_settings.languages[self.lang]['start-url-crawler']
cur_lang = languages_settings.languages[self.lang]['language']
other_langs = languages.languages.copy()
other_langs.remove(cur_lang)
other_langs = [(lang+'_').replace(' ','_') for lang in other_langs]
deny_rules = ['Category:Terms_derived_from_'+languages_settings.languages[self.lang]['language'],'Grammar']
deny_rules.extend(other_langs)
self.rules = (
Rule(link_extractor=LinkExtractor(
    restrict_css=[
        'div.CategoryTreeItem',
        'div.mw-parser-output > ul a[title^=\'Category:\']'
    ],
    deny=deny_rules
    )
),
Rule(link_extractor=LinkExtractor(
    restrict_xpaths=[
    '//*[@lang = \''+self.lang+'\' and (not(ancestor::*[@class=\'p-form-of\']) and not(ancestor::*[@class=\'form-of\']) and not(ancestor::*[@class=\'NavFrame\']))]'
    ],
    restrict_css=[
        'div.mw-category-group',
        'td#oldest-pages',
        'td#recent-additions']), callback='parse_word_page', follow=True),
)
super(dictSpider, self)._compile_rules()
```

Fonte: Produção do autor

A primeira regra serve para páginas de tópicos e categorias, tais páginas possuem links para outros endereços que não se quer realizar o *parsing*, mas ainda se deseja seguir para seus links. Por exemplo, na Figura 22, há subcategorias em uma página, tais subcategorias não são páginas a se extrair dados, mas é importante segui-la, pois eventualmente serão alcançadas páginas para extração. Nesta primeira regra, também, há uma restrição, definida no atributo “deny”, para não entrar em páginas de tópico para palavras derivadas da língua extraída em questão, isso é evidente pois os termos dentro de tal categoria serão palavras de *outros* idiomas que tem como origem alguma palavra da língua extraída em questão.

A segunda regra pegará links que levam até páginas de extração, em especial a última que restringe seguir link apenas dentro de tags que tenha como o atributo *lang* igual ao do idioma extraído. Essa regra, então, possui um callback para a função *parse_word_page* que é onde a extração dos dados ocorrem, também é colocado *follow* como *true*, isso porque, por padrão, é colocado em falso quando há um callback, assim após a extração dos dados, a aranha não seguiria os links encontrado na página de extração, por isso não se deve deixar em *false*. Por último, nesta segunda regra também foi adicionado um seletor XPath. Similarmente à seleção css, são selecionados elementos que tenham o atributo *Lang* do idioma que se deseja extrair, contudo, nesta seleção é assegurado que o elemento **não** deve possuir nodos ancestrais pertencentes a nenhuma das três classes: *p-form-of*, *form-of* ou *NavFrame.* Esta restrição existe para os dados ficarem mais limpos e com menos redundâncias.

Figura 22 - Subcategorias Wiktionary



Fonte: Tela capturada pelo autor[12]

As duas primeiras classes, *p-form-of* e *form-of*, são usadas em tags que possuem alguma forma de uma palavra (seja plural, conjugação de verbo, etc.), e a última é usado nas tabelas de conjugações de verbos. Assim evitando redundâncias de uma mesma palavra, em conjugações diferentes, aparecendo múltiplas vezes em diversas entradas do banco de dados de saída.

6.1.3.3 Pipelines

Como já comentado, o DSPD passa os itens extraídos por dois pipelines que trabalha sobre esses dados. O primeiro tem a função de limpar sujeiras que eventualmente surgem das páginas do Wiktionary, e o segundo para inserir os itens em um SQLite.

O primeiro pipeline, chamado simplesmente de *DictBotPipeline*, trata de remover resultados com eventuais erros existentes em páginas do Wiktionary, como o "*Lua error: not enough memory"*, que é um erro decorrente da linguagem de script Lua, utilizada no Wiktionary[13], essa mensagem de erro apareceu em algumas entradas do dicionário nas primeiras fases de desenvolvimento do DSPD, então esse pipeline foi criado. Outrossim, alguns outros erros foram tratados, como erros de tradução, em que uma palavra não possuía tradução, e uma mensagem de erro aparecia nos resultados[14]. Além de erros, algumas limpezas de dados são realizadas, tal como remover informações clonadas que ocorriam dentro do campo de definições.

---

12 Disponível em: <https://en.Wiktionary.org/wiki/Category:Regional_Portuguese> Acesso em: 08/03/2022

13Disponível em: <https://en.Wiktionary.org/wiki/Wiktionary:Lua_memory_errors> Acesso em: 21/06/2022

14Disponível em: <https://en.Wiktionary.org/wiki/Template:rfdef> Acesso em: 21/06/2022

O segundo pipeline trata da manipular um SQLite. Se o banco de dados não existir no momento da instanciação do pipeline, ele trata de criá-lo. Ao receber um item, já limpo pelo primeiro pipeline, será realizada a inserção no banco, de acordo com o esquema já definido na seção “6.1.2.2 Formato dos dados”.

6.1.3.4 Limitações

Anteriormente, mostrou-se como a extração dos dados é realizada e discutiu-se quais informações eram relevantes para se extrair de uma página. Contudo, é possível observar que foi utilizado o domínio em Inglês do Wiktionary, sendo possível extrair definições apenas para o inglês, e.g. português para inglês, espanhol para inglês, inglês monolíngue, etc. Porém, o HTML do Wiktionary é bem padronizado entre todos os domínios, possibilitando que se faça uma extensão para os outros domínios. O Apêndice A trata de um exemplo em que se expande o DSPD, no mínimo necessário, para pegar dados do domínio em português do Wiktionary.

Outra limitação é que, apesar de ter sido definido uma regra para evitar que se coletem dados redundantes como plurais ou conjugações, não é impedido com 100% de eficácia que algumas sujeiras sejam extraídas. Como é possível observar na Figura 23, um dicionário de português para inglês extraído capturou formas conjugadas do verbo julgar.

Figura 23 - Dados redundantes no resultado final

Fonte: Produção do autor

Isto ocorre pois tais entradas não estavam dentro de um dos nodos definidos anteriormente em que a entrada foi restringida, mas sim dentro de algum outro em que *normalmente* possui entradas que se deseja extrair.

6.1.3.5 Modo de uso

Para a execução do DSPD, faz-se necessário uma forma de salvar o progresso da extração, visto que a execução só acabaria quando todas as páginas de um idioma, alcançáveis pelas regras definida, fossem navegadas, o que pode ser muito longo. O próprio framework possui um sistema de salvar progresso, chamado de *jobs*. Basta definir um diretório para o job na hora de rodar o

scraper que todo progresso será salvo neste diretório, e carregado dele em caso de já existir progresso salvo na hora da execução.

Também é necessário passar por linha de comando alguns argumentos de configuração, como o código ISO de dois dígitos do idioma de que se deseja extrair as definições, o caminho para o banco de dados que serão armazenados os dados extraídos, e por último, na versão mais recente do DSPD, vide Apêndice A, também é necessário passar o código do idioma *para o qual* será feita a tradução ou definição das palavras.

Um exemplo de linha de comando para rodar a extração de um dicionário de inglês para português ficaria como no trecho de código a seguir:

```
scrapy crawl dict_bot -O en_ptDict/en_ptDict1.json -a db_name="en_ptDict/en_ptDict" -a lang=en
-a base_lang=pt -s JOBDIR=crawls/dictBot_en_pt-1
```

O argumento "*lang=en*" indica que o idioma a ser extraído é o inglês**, "***base_lang=pt*" indica que a extração será feita *para* o português, "*db_name="en_ptDict/en_ptDict"*" indica o caminho e nome ao banco de dados, e JOBDIR serve para apontar onde se deve salvar o progresso (job) realizado na extração. A opção "*-O"* é para gerar uma saída, neste caso em JSON.

Também é possível mexer nas configurações do DSPD no arquivo *settings.py*, contudo é recomendado que não se remova completamente o tempo de atraso das requisições, pois isso pode gerar irritação do servidor do Wiktionary, correndo riscos, em casos mais graves, de banimento de IP.

6.2 INTLIN

O INTLIN é o produto principal proposto por esse trabalho, ele surge com a intenção de integrar diferentes recursos usados em estudos de idiomas, como já discutido na seção de introdução. Seu uso é pensado para estudantes de nível intermediário em um idioma, o qual já possuem um entendimento da língua estudada, e desejam ampliar seu vocabulário ou observar a língua sendo usada em contextos mais realistas, por meio de alguma mídia de seu interesse, tendo acesso a um sistema de flashcards e dicionários em tempo real durante a utilização da aplicação, sem necessitar parar de consumir a mídia estudada, de modo a otimizar o tempo de estudo.

No INTLIN é utilizada uma arquitetura MVC para que haja separação de responsabilidades. A parte da *view* contém tudo relacionada a GUI, i.e., as especificações em FXML dos fragmentos das interfaces gráficas e seus *viewController*, que são controladores da interface gráfica usado pelo JavaFx, eles são responsável por manter variáveis de estado, ações de botões, tratamento de eventos ou qualquer lógica necessária a interface gráfica. Cada fragmento da view

pode ter no máximo um controlador. Para o *controller* têm-se controles mais gerais, que mantém as configurações gerais do sistema, variáveis de estados globais e também é responsável por se comunicar com o modelo para conseguir resultados requisitados pelo usuário na view. O modelo possui as lógicas para os módulos de dicionários e sistema de repetição espaçada.

6.2.1 Model

O Modelo do sistema contém a lógica principal da aplicação, fazendo a comunicação e manipulação de bancos de dados, por meio das lógicas apropriadas. O modelo do sistema comporta dois grandes módulos, o módulo de dicionários e o módulo de repetição espaçada (ou módulo SRS).

Ambos os módulos tiveram grande parte do seu desenvolvimento utilizando TDD, visando alcançar soluções de qualidade cobertas por testes. Sendo os testes executados sempre que alguma mudança fosse feita no modelo.

6.2.1.1 Modulo de dicionários

O módulo de dicionário contém as lógicas do dicionário, é usado principalmente para consultas de palavras, mas também há funcionalidades de alterações de dados no banco de dados dos dicionários. Também é responsabilidade desse módulo o parsing de arquivos para inserção em banco de dados SQLite. Este módulo é projetado de modo a ser expansível para que outros esquemas de banco de dados de dicionários possam ser utilizados no futuro.

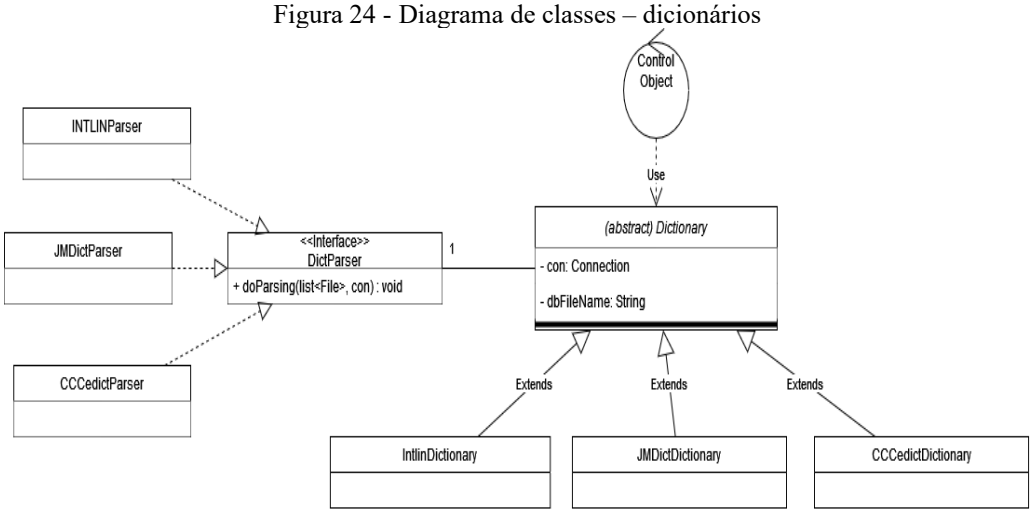
6.2.1.1.1 Arquitetura do modelo dos dicionários

Todo tipo de dicionário no sistema deve estender uma classe abstrata chamada de *Dictonary*, ela define uma API comum para os dicionários, bem como provê métodos genéricos para conexão com os bancos SQLite. A ideia é deixar o sistema expansível para que outros tipos de dicionários, com esquemas de dados diferentes, possam ser implementados no futuro, bastando que esses novos dicionários sigam a API mínima definida. Todo dicionário necessita também de um parser que converte seu arquivo original para um Banco de dados que pode ser usado pelo sistema, para o caso do novo dicionário só possuir algum arquivo, como JSON ou XML.

A Figura 24 abaixo contem uma visão simplificada de um diagrama de classes do modelo, tendo os métodos das classes de dicionários ocultados. A classe abstrata *Dictionary* contém métodos genéricos para realizar conexão com um SQLite, e uma gama de métodos abstratos a serem estendidos pelas classes filhas, que são métodos mínimos que qualquer outro tipo de dicionário que

for integrado ao sistema deve contemplar, como procurar definições e realizar alterações na base de dados.

Figura 24 - Diagrama de classes – dicionários



Fonte: Produção do autor

A classe de *Dictionary* é composta por um *parser*, o *DictParser*, que é uma interface que expõe apenas uma funcionalidade, a de realizar o parsing sobre um arquivo. A ideia desse parsing é para transformar algum tipo de arquivo de dicionário em um banco de dados que pode ser usado por uma classe de dicionário, para caso esse banco de dados não exista. Por exemplo, têm-se vários arquivos JSON gerado pelo DSPD, mas por algum motivo o banco de dados se perdeu ou não foi gerado na hora da extração, o *IntlinParser* converterá esses arquivos JSON para SQLite.

Também é possível perceber na Figura 24 que, além do dicionário padrão do INTLIN (que usa o mesmo modelo de dados do DSPD), existem dois outros tipos de dicionário, o CC-cedict (Mandarim) e o JMdict (Japônes), estes dicionários não estão completamente integrados ao sistema, não sendo possível utilizá-los no protótipo, contudo seus parses foram implementados e suas implementações serão discutidas no Apêndice B, que discute como expandir o modulo de dicionário do INTLIN.

6.2.1.1.2 Testes unitários

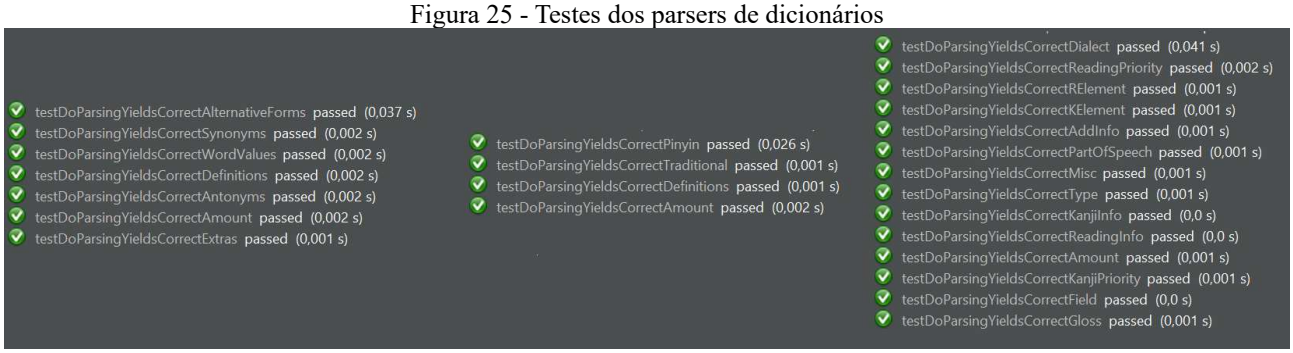
Como mencionado, o desenvolvimento deste módulo foi primariamente desenvolvido utilizando TDD. Assim, implementou-se diversos testes unitários antes mesmo de desenvolver os métodos em si.

Primeiramente, trabalhou-se nos parsers dos dicionários. Seus testes tratavam de averiguar a corretude dos dados adicionados ao SQLite, sendo o comando para realizar o parsing na

inicialização da classe de testes. Na sequência, o mesmo processo seguiu para as classes dos dicionários em si, tendo seus testes cobrindo se resultados de pesquisas eram corretos ou se alterações no banco eram feitas apropriadamente.

As três classes de parsers, explicitadas na Figura 24, tem seus testes definidos na Figura 25, sendo a primeira coluna, os testes do parser do INTLIN, a segunda do CC-cedict e a terceira do JMdict. Os testes unitários são voltados a verificar se dado elemento foi analisado e inserido em um banco de dados corretamente pela função doParsing, tendo a forma *"testDoParsingYieldsCorrect[field]"*.

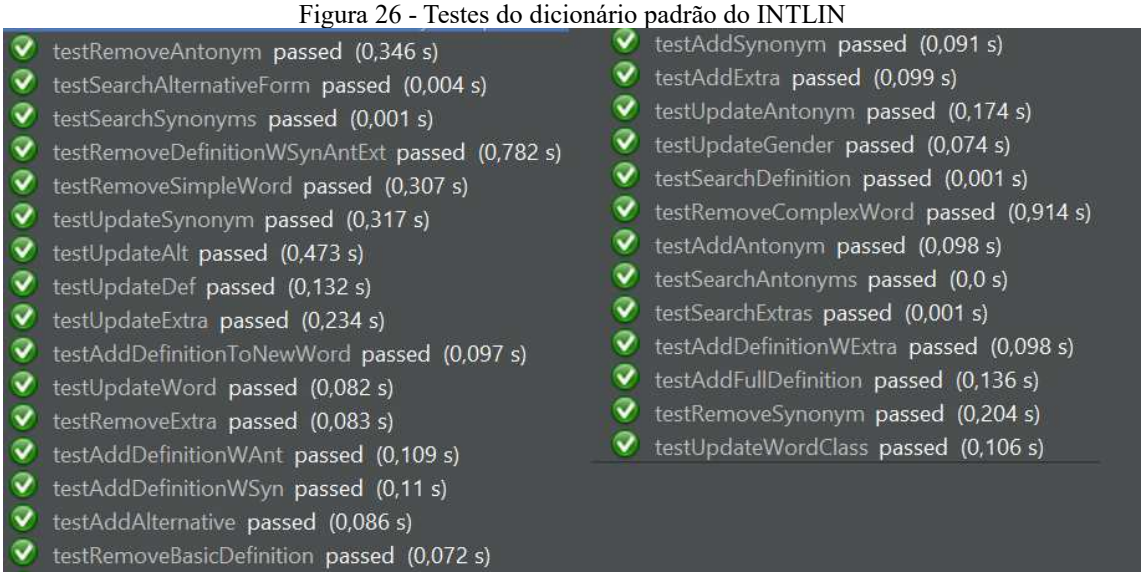
Figura 25 - Testes dos parsers de dicionários



Fonte: Produção do autor

Para o dicionário padrão do INTLIN, os testes definidos são para cobrir se cada uma das funções realiza suas ações apropriadamente, sendo alguns métodos testadas mais de uma vez para averiguar casos diferentes. A Figura 26 mostra os testes definidos para o dicionário do INTLIN.

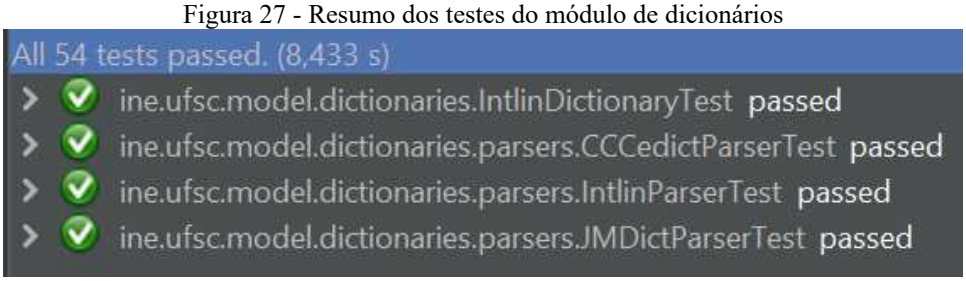
Figura 26 - Testes do dicionário padrão do INTLIN



Fonte: Produção do autor

Por fim, a Figura 27 mostra um resumo de todos os testes rodados para o módulo de dicionários. Este modulo resultou em um total de 54 testes de unidade cobrindo diversas ações do dicionário INTLIN, bem como a corretude dos parses do INTLIN, JMdict e CCcedict.

Figura 27 - Resumo dos testes do módulo de dicionários



Fonte: Produção do autor

6.2.1.2 Modulo de SRS

Este módulo é responsável pela criação e administração dos flashcards, tendo a responsabilidade de definir datas de revisões de cartões estudados de acordo com o feedback do usuário e do nível de proficiência com dado cartão.

O sistema para repetição espaçada usado no INTLIN é baseado no *Sistema de Leitner*, que é um algoritmo que utiliza cinco grupos de proficiência no qual os cartões podem se enquadrar. Quanto mais proficiência um usuário tiver em um cartão, mais espaçadas as revisões desse cartão serão.

No INTLIN, quando o usuário revisa seus cartões, ele pode responder que o conteúdo de um cartão foi: (i) Impossível de lembrar/falhou. (ii) Difícil de lembrar. (iii) Conseguiu lembrar. (iv) Fácil de lembrar.

De acordo com seu nível de proficiência no cartão e com sua resposta, uma data de próxima revisão será calculada. Para caso de falha, isto é, a não lembrança do conteúdo do cartão, a próxima revisão é agendada para o dia atual e o nível de proficiência é reiniciado para o mais baixo. Respostas difíceis não causam mudança de proficiência mas geram uma data de revisão próxima do dia atual. Respostas boas aumentam a proficiência por um nível e agendam a revisão para datas um pouco mais espaçadas. Por último, respostas fáceis aumentam proficiência por dois níveis e geram repetições bem mais espaçadas.

Também existe um fator de *ease*, que é aumentado com respostas fáceis ou, dependendo da proficiência, boas. Esse fator pode dar uns dias extras no agendamento da revisão, caso esteja em

um valor alto o suficiente. Ele existe para que em caso de errar um cartão, não gere uma resposta tão punitiva para quem no passado já o tinha aprendido.

No calculo para a agendar uma data de revisão é utilizado o padrão de projeto *estratégia*, no qual a estratégia para o cálculo é escolhida baseada na dificuldade informada pelo usuário para o cartão.

Figura 28 - Padrão estratégia no cálculo da data de revisão de um cartão

```
private LocalDate calcBaseReview(Difficulty difficulty) {
    CalcReviewStrategy reviewCalculator = selectStrategy(difficulty);
    ease = reviewCalculator.updateEase();
    level = reviewCalculator.updateProficiency();
    nextReview = reviewCalculator.calcNextReview();

    return nextReview;
}

private CalcReviewStrategy selectStrategy(Difficulty answer) {
    switch (answer) {
        case easy:
            return new EasyReviewStrategy(level, ease);
        case good:
            return new GoodReviewStrategy(level, ease);
        case hard:
            return new HardReviewStrategy(level, ease);
        default:
            return new FailReviewStrategy(level, ease);
    }
}
```

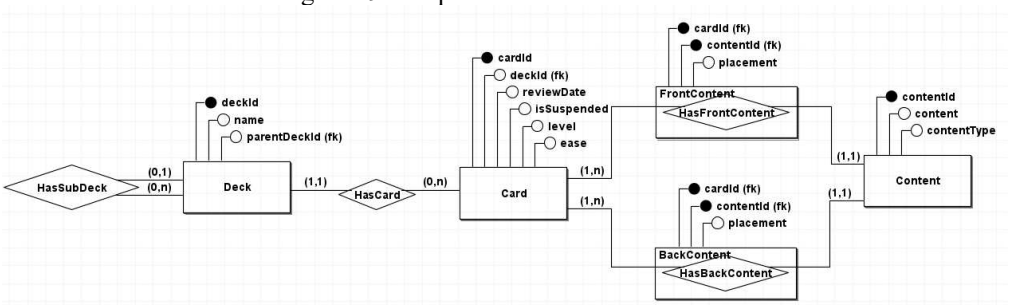
Fonte: Produção do autor

6.2.1.2.1 Estrutura do sistema SRS

Similar aos dicionários, os cartões do usuário são mantidos localmente em um SQLite. Seu esquema é apresentado pela Figura 29.

Em suma, os cartões possuem as informações principais para a lógica de calcular prazos de revisões: level (proficiência), ease e a data de revisão atual em si. Os cartões podem ter múltiplos conteúdos na frente e no seu verso, e cada conteúdo pertence unicamente a um cartão, isso pode ser observado pelas cardinalidades na tabela associativa, um cartão pode estar várias vezes dentro de tais tabelas, enquanto cada conteúdo aparecerá uma única vez, i.e. junto ao seu cartão. Cada cartão pertence unicamente a um baralho, que por sua vez pode ser um “sub-deck” de outro baralho, e.g. o baralho de “conjugações de verbos” é um sub-deck do baralho “gramática”. Os atributos placement das tabelas associativas e contentType de content servem para controles dentro da aplicação.

Figura 29 - Esquema Banco de dados SRS



Fonte: Produção do autor

Os cartões a serem revisados no dia são carregados em memória em uma classe do modelo, onde podem ser calculadas todos os pontos relacionados ao próximo agendamento de revisão, já mencionados na seção 6.2.1.2 Modulo de SRS.

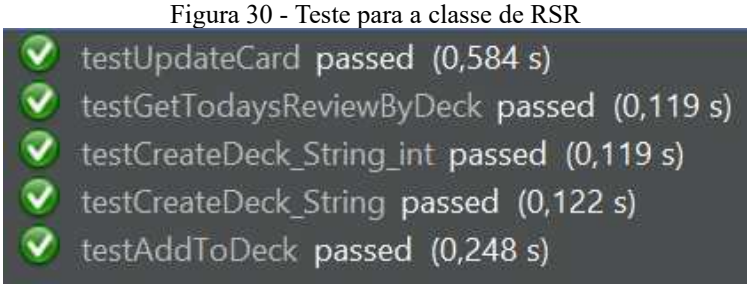
6.2.1.2.2 Testes unitários

Igualmente ao módulo de dicionários, este também foi desenvolvido por uma abordagem dirigida a testes. Foram testadas duas classes dentro deste módulo, a classe do cartão em si e a classe do RSR, que é responsável por comunicar com o banco de dados.

Para a classe do SRS, foram definidos 5 testes, que verificam se as revisões do dia foram pegas corretamente, atualização e adição de cartão em um baralho e criação de baralhos. A Figura 30 explicita estes testes.

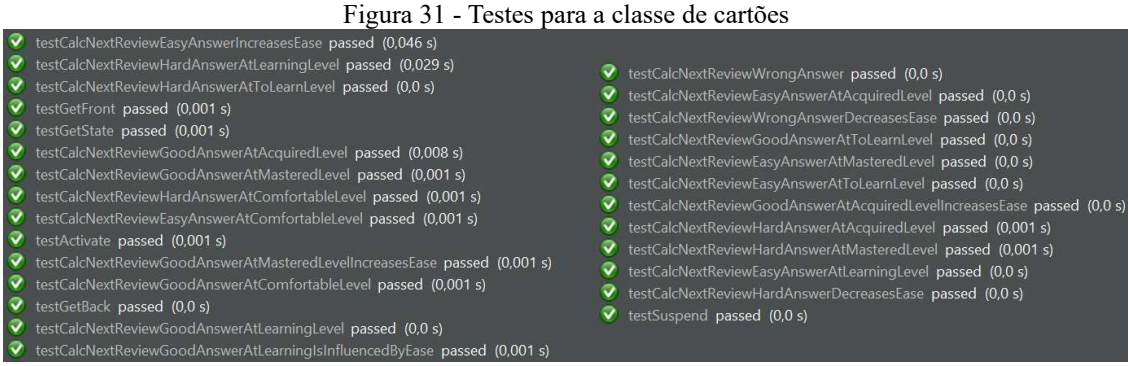
Para a classe dos cartões foram 27 testes, em sua grande maioria testando combinações de resposta para um cartão com nível de proficiência do mesmo, vide Figura 31.

Figura 30 - Teste para a classe de RSR



Fonte: Produção do autor

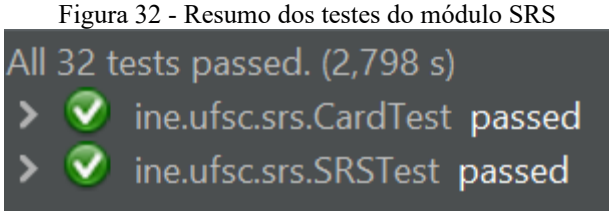
Figura 31 - Testes para a classe de cartões



Fonte: Produção do autor

A Figura 32 traz o relatório da execução dos testes deste módulo, foram ao total 32 testes desenvolvidos para cobrir majoritariamente se as revisões estão corretas, mas também foram testados funcionalidades relacionada ao banco de dados, tais quais adicionar cartão a um baralho, criar baralho, atualizar um conteúdo de algum cartão, entre outros.

Figura 32 - Resumo dos testes do módulo SRS



Fonte: Produção do autor

6.2.2 Controller

A parte de controladores do sistema dispõe de dois controladores globais, um controlador principal que administra informações gerais de estado do sistema, e um controlador de configurações que é responsável por trabalhar com o arquivo de configuração.

6.2.2.1 Controlador global

O Controlador global é uma classe *singleton*, que é majoritariamente chamada pela camada de visão (View) para que alguma requisição ao modelo seja feita. Bastante informação de estado do sistema também são salvas nesta classe: idiomas suportados, idioma selecionado, dicionários carregados, listas de cards para revisão do dia, entre outros.

Por muitas dessas variáveis serem do estado global do sistema, alterá-las pode implicar que seja necessário executar códigos em outras camadas da aplicação. Por isto, é utilizado no controle o

padrão o*bserver*, no qual o controlador global implementa uma interface chamada "*Observable*" e poderá notificar classes observadoras que se inscrevam no controlador por meio do método *attach*.

Figura 33 - Padrão observer no controlador global

```
public class Controller implements Observable {

    public static Controller instance = new Controller();

    private final List<Observer> observers = new ArrayList<>();

    @Override
    public void attach(Observer o) {
        observers.add(o);
    }

    @Override
    public void detach(Observer o) {
        observers.remove(o);
    }

    @Override
    public void notifyUpdate(Message message) {
        observers.forEach(observer -> {
            observer.update(message);
        });
    }
```

Fonte: Produção do autor

De modo geral, os observadores serão *viewControllers*, e as notificações serão usadas para atualizar a interface gráfica de algum modo.

6.2.2.2 Controlador de configurações

O sistema utiliza um arquivo de configurações em XML, que, no momento, mantém duas configurações: uma data (usada para uma lógica interna) e o último idioma carregado para estudar. Este controlador trabalha com leitura e escrita do arquivo de configurações, utilizando a classe utilitária Properties, nativa do Java.

O papel deste controlador é pequeno, sendo utilizado para recuperar e escrever informações de configurações no arquivo, também é usado na inicialização da aplicação, quando é lido o arquivo de configuração e salvas todas as informações em memória.

6.2.3 View

As interfaces gráficas do sistema foram implementadas com a plataforma JavaFx, junto a ferramenta SceneBuilder[15] que permite a fácil construção de layouts via *drag and drop*, evitando trabalhar diretamente com o arquivo estilo XML que o JavaFx utiliza. Esta seção destaca os principais pontos de desenvolvimento do protótipo.

Inicialmente, adentrar-se-á em questões de design definidas para a construção da interface visual. Por último, essa seção apresentará tudo o que foi construído como parte visual para o protótipo.

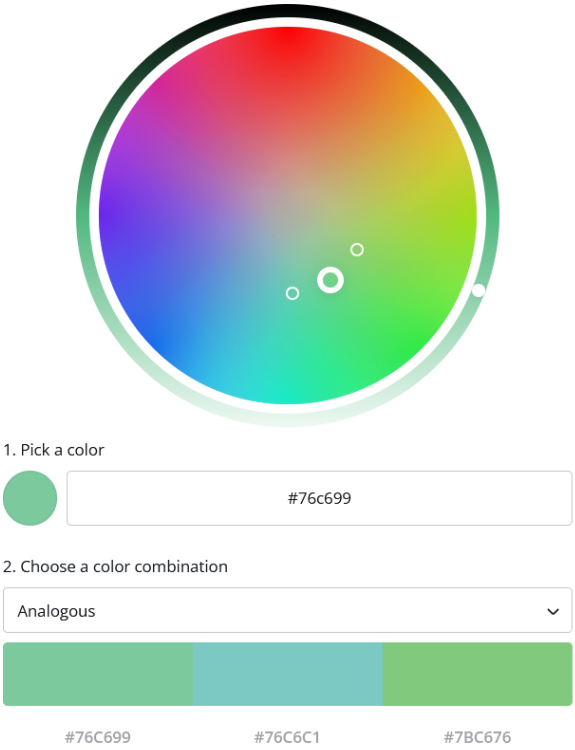
6.2.3.1 Escolhas de estilos e design

Para garantir que o protótipo fique minimamente consistente em termos visuais, definiu-se previamente um esquema de cor e iconografia que seriam necessários no protótipo, porém não foi realizado nenhum estudo mais aprofundado para embasar as escolhas realizadas, e sim foram escolhidas para garantir uma consistência visual miníma. É importante explicitar que as escolhas para interface gráfica não foram feitas junto a um designer, sendo a interface gráfica definida apenas para poder validar as ideias do protótipo, contudo se tentou tomar um certo cuidado com aspectos de usabilidade.

Para a paleta de cores, em adição as cores neutras como preto, branco e cinza, escolheu-se uma paleta de cores análogas, isto é, três cores próximas na *color wheel*, sendo uma cor dominante, e outras duas auxiliares, para destaque de informações. A Figura 34 ilustra a seleção de cores para o sistema, começando pela primária, e seguindo para as auxiliares. A escolha para essas cores não seguiu um padrão muito técnico, foi escolhida uma cor primária que tivesse um bom contraste com o preto, para poder colocar texto sobre. A partir dessa escolha, selecionou-se as análogas, que também possuem bom contraste com o preto. Para verificar o contraste, rodou-se um teste que verifica adesão as recomendações WCAG, Diretrizes de Acessibilidade para o Conteúdo da Web. O resultado dos testes está exposto na Figura 35.

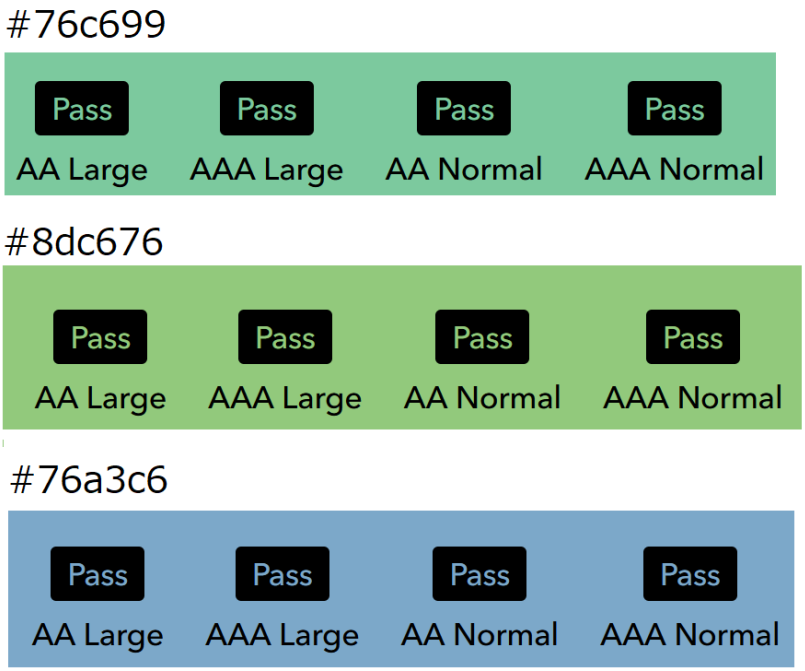
15 Disponível em: <https://gluonhq.com/products/scene-builder/>

Figura 34 - Paleta de cores análogas



Fonte: Tela capturada pelo autor[16]

Figura 35 - Testes de contraste



Fonte: Tela capturada pelo autor[17]

16 Disponível em: <https://www.canva.com/colors/color-wheel/> Acesso em: 10/03/2022

17 Disponível em: <https://colourcontrast.cc/> Acesso em: 20/05/2022

Também utilizou-se um conjunto de seis ícones disponíveis para uso gratuito, dadas as devidas atribuições, do Icons8[18]. Escolheu-se apenas ícones contendo bordas com algum nível de arredondamento, a fim de garantir uma maior consistência visual. Idealmente, os ícones deveriam ter o formato vetorizado, para que fosse possível alterar tamanhos, sem comprometimento na qualidade da imagem, ícones como vetores também haveria a possibilidade de trocar as cores por demanda, contudo os ícones vetorizados do Icons8 só estão disponíveis sob a licença paga. Deste modo, optou-se por utilizar imagens em png, escolhidas nos tamanhos e cores apropriados, sendo as cores utilizadas as mesmas da paleta de cores escolhida juntamente ao preto e o branco. Tais ícones utilizados na aplicação estão apresentados na Figura 36.

Figura 36 - Ícones do sistema

Fonte: Produção do autor[18]

6.2.3.2 Apresentação da GUI

A interface gráfica do sistema será apresentada nesta seção do trabalho, começando por uma visão geral de como as informações são organizadas no sistema, e então se adentrará especificidades de como a interface trabalha com os dicionários, as mídias e com o sistema de repetição espaçada.

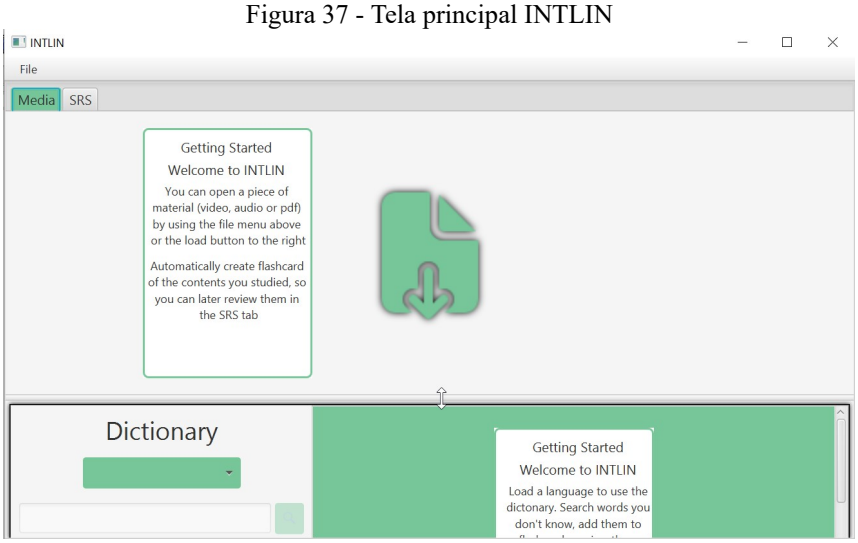
6.2.3.2.1 Visão geral

Primeiramente, desenvolveu-se a tela inicial, tendo como principal consideração que o sistema de dicionários deve estar acessível durante o estudo da mídia. Desta forma, os dicionários ficam dispostos no inferior da tela em uma área expansível, deste jeito, é possível esconder os dicionários durante o estudo de algum conteúdo, e trazê-los à vista quando houver necessidade. Acima desta região expansível dos dicionários, está disposto um painel de *tabs*, em que é possível alternar entre o módulo de mídia e do sistema de SRS, o motivo de tal separação se dá pelo fato que, de modo geral, os dois não serão usados ao mesmo tempo, sendo a aba de mídia usada para estudos e a de SRS para revisões. A Figura 37 ilustra o descrito. Assim que o sistema é aberto, tem-

18 Disponível em: <https://icons8.com.br/> Acesso em: 10/03/2022

se na aba de mídias um botão em formato de ícone para carregar um arquivo, além disso, não há muita informação para apresentar na tela, então visando aproveitar um pouco do espaço disponível, foram inseridos regiões com informações rápidas sobre o uso do sistema.

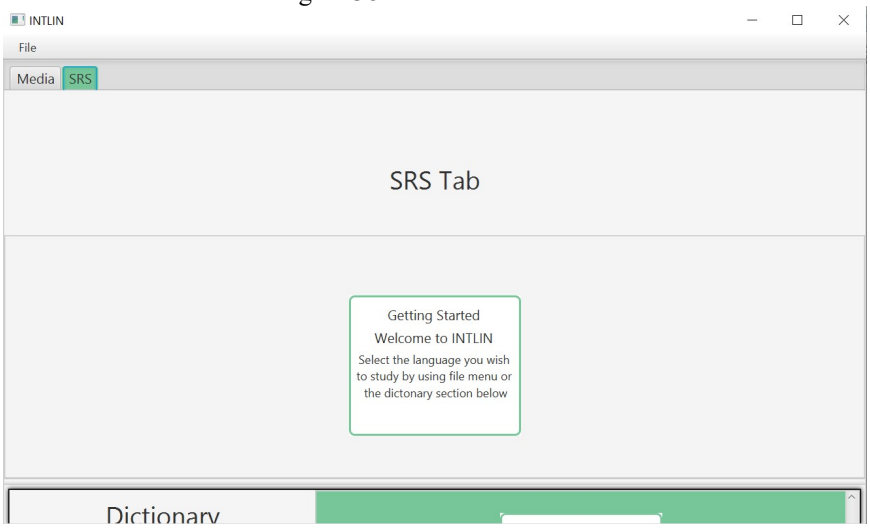
Figura 37 - Tela principal INTLIN



Fonte: Produção do autor

A aba de SRS, também não contem nenhuma informação para apresentar quando não há um idioma carregado para estudo no sistema, então para ocupar a área inutilizada, também foi inserido um *helper text*, como possível visualizar na Figura 38.

Figura 38 - Aba de SRS vazia



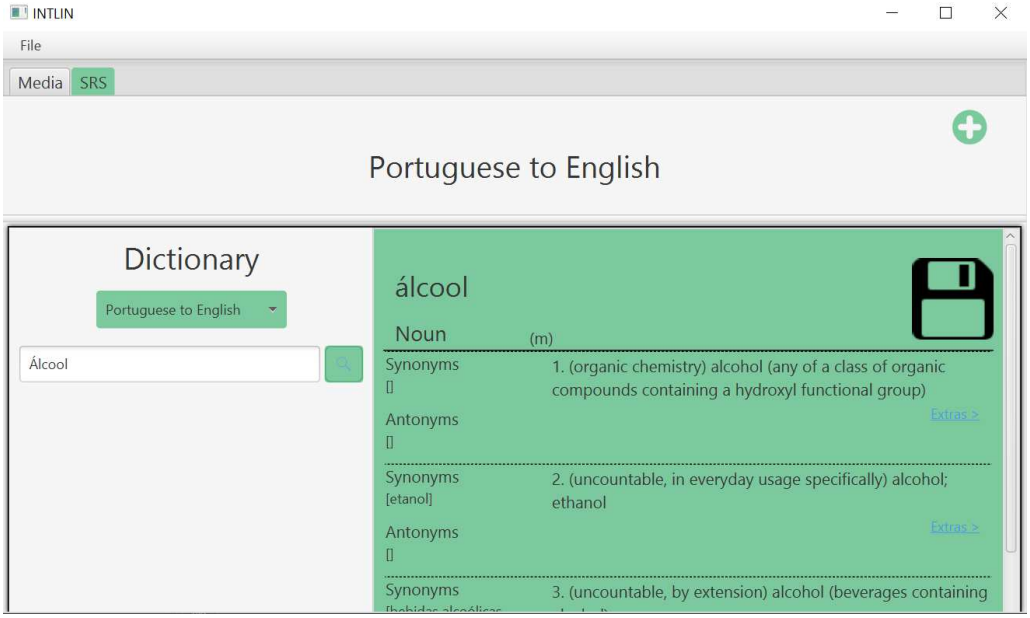
Fonte: Produção do autor

6.2.3.2.2 Região do dicionário

Para a região expansível de dicionários, foi feita uma divisão em duas colunas, a primeira é responsável por selecionar o dicionário e realizar consultas, enquanto a segunda tem a responsabilidade de apresentar o resultado da busca. É averiguável pela Figura 37, que quando não há um idioma carregado, o campo de busca e seu botão ficam desabilitados. É possível escolher um idioma pelo menu *file* ou pela caixa de seleção na região de dicionários, após selecionar o idioma desejado, é possível realizar consultas no dicionário.

A Figura 39 apresenta uma consulta realizada para a palavra "Álcool" em um dicionário de português para Inglês. Também é possível observar, um botão para salvar uma entrada como um flashcard em um baralho de revisão, bem como regiões para os outros dados que foram extraídos do Wiktionary, tais quais sinônimos e antônimos.

Figura 39 - Pesquisa no dicionário



Fonte: Produção do autor

6.2.3.2.3 Aba de mídias

Na aba de mídias, é apresentado um ícone clicável em formato de arquivo, como já visto na Figura 37. A partir dele, é possível abrir arquivos de áudio, vídeo ou PDF. Carregar áudio ou vídeo levará a uma mesma tela, sendo a diferença que para áudio, onde seria exibido o vídeo, é apresentado o segundo ícone da Figura 36 centralizado. PDF abre uma webview junto com um plugin de visualização de PDF.

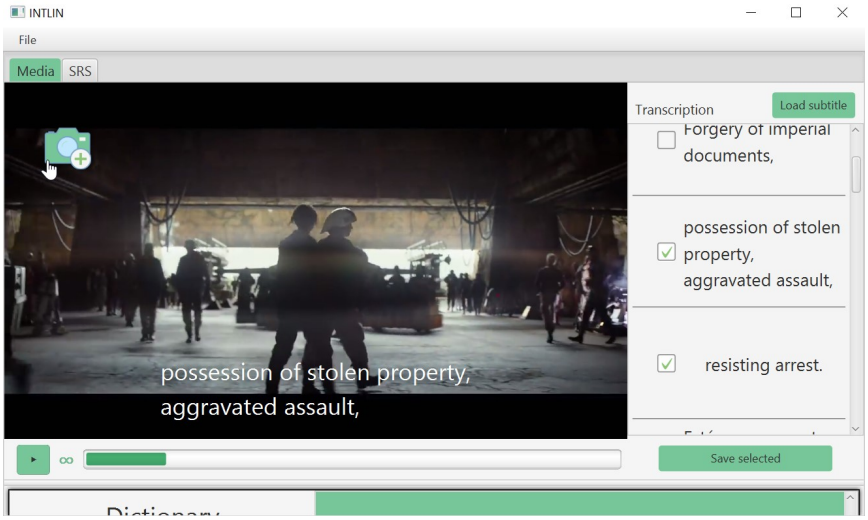
Ao carregar áudio ou vídeo, duas seções são renderizadas na tela, a principal com o vídeo e barra de progresso e uma outra onde ficará a transcrição da legenda, como podemos observar na

Figura 40. Na região da transcrição de legenda há um botão no topo para se carregar uma legenda, que deve ser um arquivo de extensão *srt*. Na parte inferior à transcrição é possível visualizar um botão desativado, ele permanecerá desativado enquanto não houver uma legenda ou um idioma carregado no sistema, e ele é utilizado para salvar trechos selecionados da transcrição de legendas em um flashcard que poderá ser eventualmente revisado pelo usuário. A Figura 41 explicita o sistema com legendas carregadas, bem como algumas selecionadas.

Figura 40 - Vídeo aberto no INTLIN

Fonte: Produção do autor

Figura 41 - Botões na aba de mídia



Fonte: Produção do autor

Em ambas as Figuras também está presente um botão de câmera sobre o vídeo, este botão realiza uma captura de tela do vídeo e adiciona em um flashcard. Na primeira Figura, ele se encontra levemente transparente, isso ocorre para não atrapalhar a visualização da mídia em

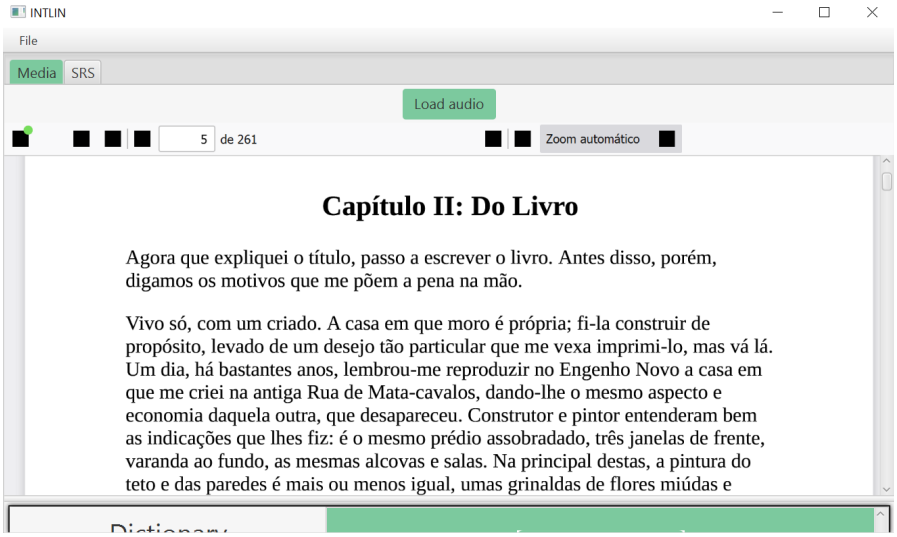
execução. Ao passar o cursor por cima, um efeito de *highlight* ocorre evidenciando o botão, como possível ver na Figura 41. Quando um dos dois botões são acionados, uma janela aparecerá pedindo para que o usuário adicione uma solução ao flashcard e confirme sua criação, vide a Figura 42.

Figura 42 - Criação de flashcard a partir de mídia

Fonte: Produção do autor

Para a exibição de PDFs foi usado uma webview do JavaFx junto com um *plugin* PDF.js[19]. A Visualização de PDF, apesar de funcional, possui alguns bugs, um visual em que os botões do leitor não aparecem corretamente renderizados, e um segundo bug mais grave de causa não identificada em que a aplicação inteira fica lenta e pode levar até a crashes no sistema. O bug visual pode ser averiguado na Figura 43, em que mostra um PDF aberto no INTLIN. Também é possível carregar um arquivo de áudio através do botão "*Load audio*", em que o sistema permite ao usuário ouvir áudio enquanto lê o texto, para funcionalidade de audiobook. Áudio e PDF podem ser vistos simultaneamente carregados na Figura 44.

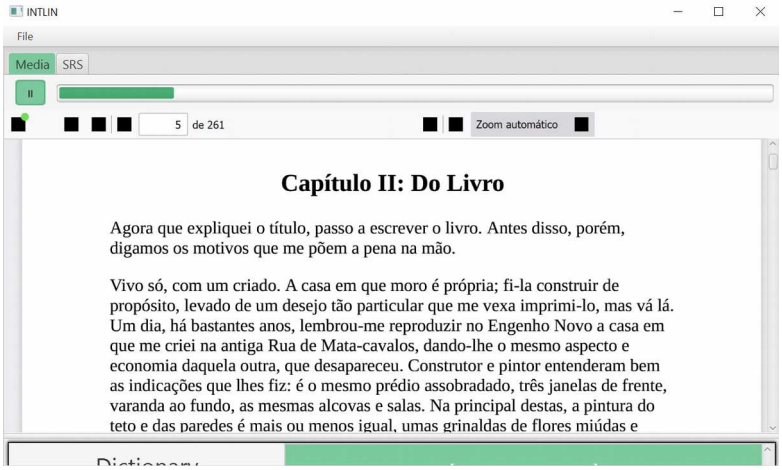
Figura 43 - PDF aberto no INTLIN



Fonte: Produção do autor

19 Disponível em: <https://github.com/mozilla/pdf.js/blob/master/LICENSE>

Figura 44 - PDF e áudio abertos no INTLIN

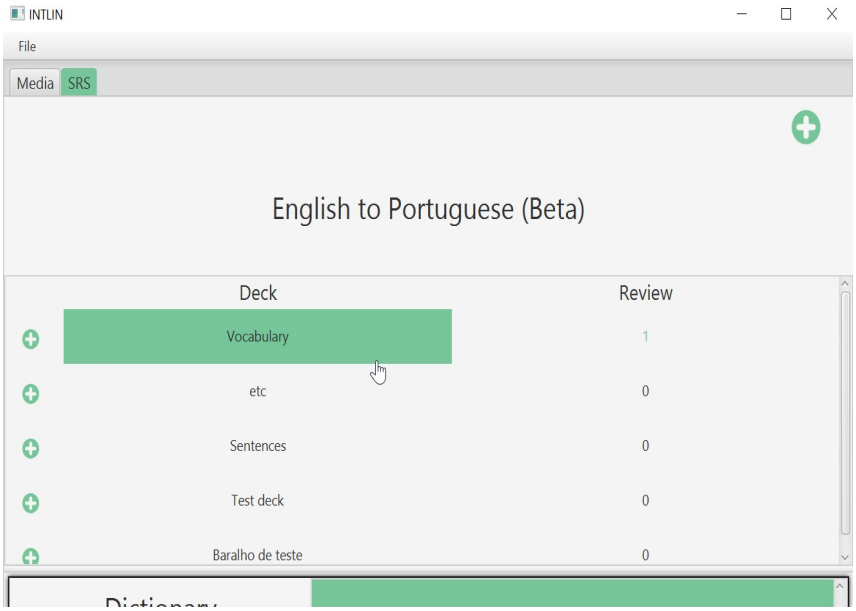


Fonte: Produção do autor

6.2.3.2.4 Aba de SRS

Por último, a aba de SRS apresenta o nome do idioma selecionado para estudos como cabeçalho, um botão para criação de novos baralhos localizado no topo, o nome de todos os baralhos existentes, para cada baralho há um botão para criação de cartões, bem como uma contagem de revisões agendadas para o dia. Vide Figura 45.

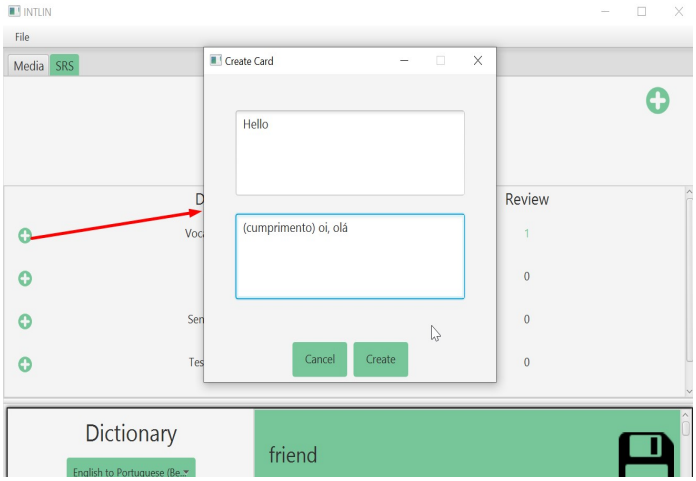
Figura 45 - Aba de SRS



Fonte: Produção do autor

Ao clicar no botão de adição ao lado de um baralho, uma janela para criação de um cartão é aberta, ao final deste processo, o cartão será adicionado ao baralho designado, a Figura 46 demonstra a criação de um cartão manualmente.

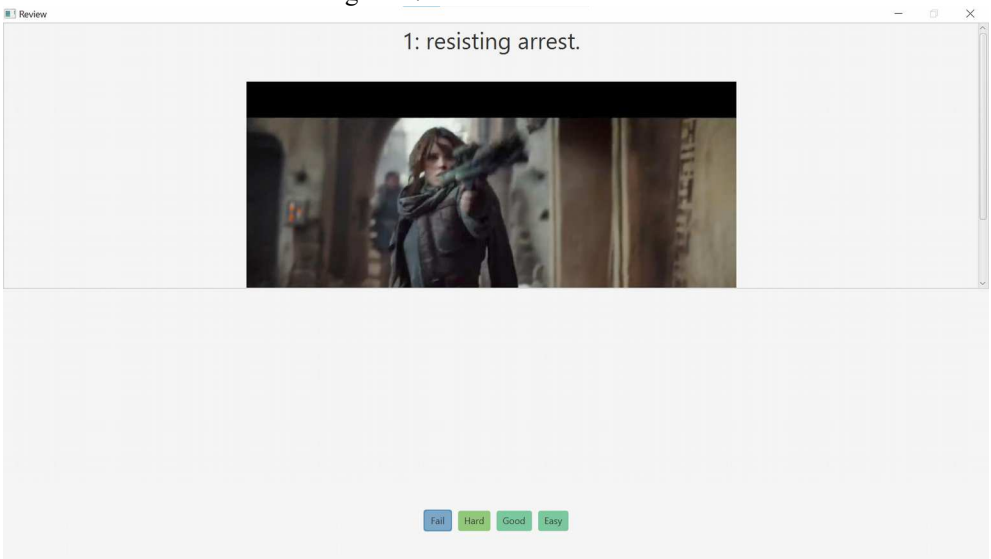
Figura 46 - Criação de cartão manualmente



Fonte: Produção do autor

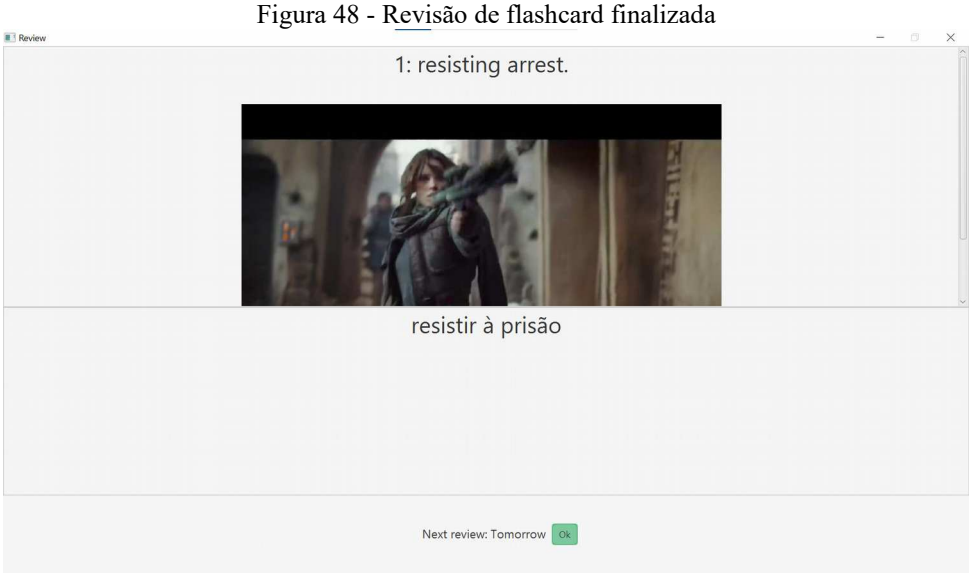
Quando o usuário clicar em um baralho, abrirá uma janela de revisões, a qual será exibido, sequencialmente, todos os cartões que o usuário tem agendado para o dia. Tais cartões são capazes de exibir textos e imagens, e podem ser criados manualmente como visto acima, via log de legendas e por screenshot de um vídeo carregado no sistema. A Figura 47 é referente a um cartão aberto para revisão, contendo ambos texto e imagem, na parte superior da tela tem o conteúdo a ser revisado, abaixo disso tem uma região escondida que aparecerá a resposta, e na parte inferior da tela estão dispostos os quatro botões para o usuário auto avaliar o quão bem ele se lembra do conteúdo. Quando a revisão de um cartão é finalizada, a resposta do mesmo é exibida e é apresentado o *feedback* da data que o cartão estará disponível novamente para estudo, vide Figura 48.

Figura 47 - Revisão de flashcard



Fonte: Produção do autor

Figura 48 - Revisão de flashcard finalizada



Fonte: Produção do autor

6.2.4 Limitações do protótipo

O protótipo definiu as principais funções do sistema, que tiveram seu desenvolvimento relatado na seção anterior. Contudo, ainda existem algumas funcionalidades que ficaram de fora desta primeira versão, dentro do modelo existem algumas funcionalidades implementadas que não foram portadas na interface gráfica do sistema, sendo impossível o usuário realizar tais ações, e.g., alterar valores nos bancos de dados dos dicionários, suspender um cartão, criação de subdecks, dentre outros.

Dentro do protótipo, existe um bug com causa não identificada quando há um PDF carregado, como já mencionado em 6.2.3.2.3 Aba de mídias, o problema é relativamente recorrente, o que pode tornar o estudo com PDF um pouco frustrante ao usuário. Não se consegui traçar uma causa para tal bug, assim o mesmo não foi corrigido. Fora isso, as demais funcionalidades constadas no protótipo funcionam como esperado.

7 AVALIAÇÃO DO PROTÓTIPO

Após concluir o desenvolvimento do protótipo, surge a necessidade de uma avaliação com alguns usuários. Então, para a avaliar o protótipo, definiu-se um teste de usabilidade para se realizar em ambiente controlado com três participantes anônimos. Os três são bons membros do público-alvo, pois um gosta de estudar idiomas como *hobby* e os outros dois são estudantes de Inglês.

Para o teste, planejou-se um conjunto de atividades para os participantes realizarem, e para cada uma delas levantou-se questões de análise quanto à eficácia e eficiência do sistema, isto é, se a porcentagem esperada dos participantes conseguiram concluir a tarefa e se as tarefas foram concluídas dentro do tempo esperado. Após realizarem as atividades, a fim de medir o nível de satisfação dos participantes do teste, o questionário *System Usability Scale* (SUS) foi aplicado e ficou aberto um espaço para comentários sobre o protótipo.

O objetivo do teste é para que se obtenha feedback de usuários, para poder analisar se as escolhas realizadas na interface estão boas em termos de usabilidade, podendo traçar um plano de onde prosseguir e do que melhorar.

7.1 DEFINIÇÃO DO TESTE

Levantou-se 10 atividades para o teste e para cada uma delas foi definido a porcentagem de participantes que se espera que consigam concluir a atividade, bem como a quantidade máxima de tempo esperada para a conclusão. As atividades selecionadas, bem como as métricas esperadas estão dispostas na Tabela 6 abaixo. Nesta tabela, estão dispostas as atividades selecionadas para realização no teste, bem como os *requisitos* de usabilidade para eficácia e eficiência, por exemplo, para a Atividade 1 espera-se que cada um dos três participantes consiga concluí-la em um tempo de até 18 segundos.

Para medir níveis de *satisfação*, também foi realizado o questionário SUS com os participantes, cujo o objetivo é ter pontuação acima de 75 pontos, que é o mínimo considerado bom. Abaixo disto, é um forte indicativo que o design do sistema precisa ser repensado ou refatorado em alguns aspectos.

Tabela 6 - Atividades do teste

#	Atividade	Eficácia	Eficiência
1	Selecionar um idioma para estudo no sistema	3/3	18 segundos
2	Carregar uma mídia de vídeo para estudar	3/3	24 segundos
3	Carregar uma legenda para o vídeo carregado	3/3	24 segundos
4	Usar o dicionário para pesquisar alguma palavra	3/3	18 segundos

5	Salvar uma entrada do dicionário em um flashcard	3/3	9 segundo
6	Salvar uma captura de tela em um flashcard	3/3	27 segundos
7	Salvar trechos da transcrição de legenda em um flashcard	3/3	27 segundos
8	Abrir um baralho e revisar um cartão	3/3	30 segundos
9	Fechar a mídia de vídeo aberta	2/3	21 segundos
10	Carregar PDF + áudio (audiobook)	3/3	58 segundos

Para termos de eficiência, as atividades de carregamento de arquivos no sistema (2, 3 e 10) ganharam valores consideravelmente altos, pois são atividades em que se deve esperar o seletor de arquivos do sistema operacional abrir, pensar onde está o arquivo e navegar até o mesmo, a atividade 10 é o tempo de abrir dois arquivos somados com um tempo adicional para caso o usuário queira olhar o PDF antes de abrir um áudio (24 s + 24 s + 10 s). A atividade de revisão dos flashcards criados durante o teste também recebe um tempo esperado de execução elevado, levando em consideração diversos fatores, tais como o usuário terá de ler o cartão (algum pode ter bastante texto), o usuário pode marcar algum cartão como errado (Botão *fail*) múltiplas vezes, o que faria o cartão aparecer diversas vezes para revisão. Tarefas que envolvem digitar respostas de flashcards (e.g. salvar captura e trechos de legenda) receberam um tempo generoso também, caso os participantes queiram digitar alguma informação.

Quanto a eficácia, acreditava-se que a única atividade que poderia gerar algum problema seria a 9ª, quanto ao fechamento de uma mídia carregada, pois a ação para fechar uma mídia carregada não está à vista nas telas principais onde o usuário já está focado e sim está escondido como um botão de menu dentro do menu *file*, essa desconfiança por si só já pode indicar um problema com este design. Desta forma, colocou-se que um dos três participantes do teste podem vir a falhar nesta atividade.

7.2 EXECUÇÃO DO TESTE

Com cada um dos participantes, a execução do teste iniciou com a afirmação de que caso não consigam realizar algumas das atividades requisitadas a culpa não é dos participantes e sim do design utilizado no protótipo, e que esse teste é justamente para avaliar a usabilidade e qualidade do design adotado no sistema.

O teste foi realizado presencialmente, e os participantes aceitaram em realizar o teste de forma anônima e na Figura 49 é possível observar os participantes realizando o teste, de costas e borrados, sem revelar suas identidades.

Após um breve explicação sobre o sistema, seguiu-se para as atividades. As Tabelas abaixo trazem os resultados por atividade. Os valores foram preenchidos em vermelho quando não se alcançou o valor estipulado e em verde caso contrário. A Tabela 17, no final, consta os resultados obtidos no questionário SUS.

Tabela 7 - Relatório do teste: atividade 1

Selecionar um idioma para estudo no sistema	Concluiu a atividade	Tempo
Participante 1	✔	12 segundos
Participante 2	✔	10 segundos
Participante 3	✔	10 segundos

Tabela 8 - Relatório do teste: atividade 2

Carregar uma mídia de vídeo para estudar	Concluiu a atividade	Tempo
Participante 1	✔	16 segundos
Participante 2	✔	20 segundos
Participante 3	✔	11 segundos

Tabela 9 - Relatório do teste: atividade 3

Carregar uma legenda para o vídeo carregado	Concluiu a atividade	Tempo
Participante 1	✔	12 segundos
Participante 2	✔	10 segundos
Participante 3	✔	10 segundos

Tabela 10 - Relatório do teste: atividade 4

Usar o dicionário para pesquisar alguma palavra	Concluiu a atividade	Tempo
Participante 1	✔	6 segundos
Participante 2	✔	5 segundos
Participante 3	✔	9 segundos

Tabela 11 - Relatório do teste: atividade 5

Salvar uma entrada do dicionário em um flashcard	Concluiu a atividade	Tempo
Participante 1	✔	10 segundos

Participante 2	✔	11 segundos
Participante 3	✔	3 segundos

Tabela 12 - Relatório do teste: atividade 6

Salvar uma captura de tela em um flashcard	Concluiu a atividade	Tempo
Participante 1	✔	10 segundos
Participante 2	✔	12 segundos
Participante 3	✔	10 segundos

Tabela 13 - Relatório do teste: atividade 7

Salvar trechos da transcrição de legenda em um flashcard	Concluiu a atividade	Tempo
Participante 1	✔	8 segundos
Participante 2	✔	3 segundos
Participante 3	✔	8 segundos

Tabela 14 - Relatório do teste: atividade 8

Realizar as revisões dos flashcards gerados durante o teste	Concluiu a atividade	Tempo
Participante 1	✔	35 segundos
Participante 2	✔	20 segundos
Participante 3	✘	(tempo até desistência) 27 segundos

Tabela 15 - Relatório do teste: atividade 9

Fechar a mídia de vídeo aberta	Concluiu a atividade	Tempo
Participante 1	✔	26 segundos
Participante 2	✔	9 segundos
Participante 3	✔	31 segundos

Tabela 16 - Relatório do teste: atividade 10

Carregar PDF + áudio (audiobook)	Concluiu a atividade	Tempo
Participante 1	✔	37 segundos
Participante 2	✔	25 segundos
Participante 3	✔	17 segundos

Tabela 17 - Relatório do teste: questionário SUS

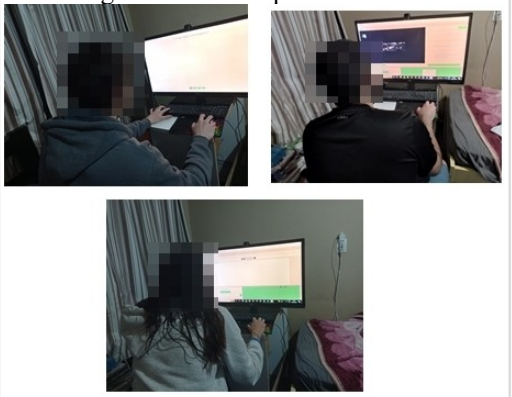
Pontuação SUS	Pontuação
Participante 1	100 pontos
Participante 2	72.5 pontos
Participante 3	75 pontos

Durante o teste, pode-se obter um feedback muito valioso dos participantes, bem como observar inconsistências e pontos fracos no protótipo. Como esperado, na atividade 9 foi difícil para todos os participantes fecharem a mídia aberta, apesar de todos terem concluído a atividade, os tempos foram consideravelmente elevados, o feedback comum também foi o esperado, a funcionalidade está escondida no menu *file*. A atividade 8 esperava-se 100% de conclusão, contudo o Participante 3 não havia compreendido que as revisões iriam para baralhos na aba de SRS, o que levou a desistência da atividade. Na atividade 5, dois dos participantes tiveram dificuldade para encontrar onde salvar uma entrada do dicionário para revisão, apesar do botão de "salvar" estar grande no topo da entrada, o feedback comum dos três é que ele não tinha "cara de botão", parecendo apenas uma imagem ou decoração (sendo assim ignorado), o Participante 2 também se questionou se a entrada foi realmente salva para revisão, pois não há nenhum feedback do sistema informando tal. A atividade 10, apesar de todos os participantes terem concluído a tempo, pode-se observar um certo *delay* para encontrar o botão "*load audio*" acima do PDF, o que pode indicar uma necessidade de realçar o botão de alguma outra maneira, de modo que fique mais visível e intuitivo para o usuário achá-lo.

Por último, o protótipo obteve 72.5 pontos no questionário do Participante 2, isso somado a todo o feedback recebido leva a entender que ainda há necessidade de aprimoramentos e refatoração em algumas partes do protótipo.

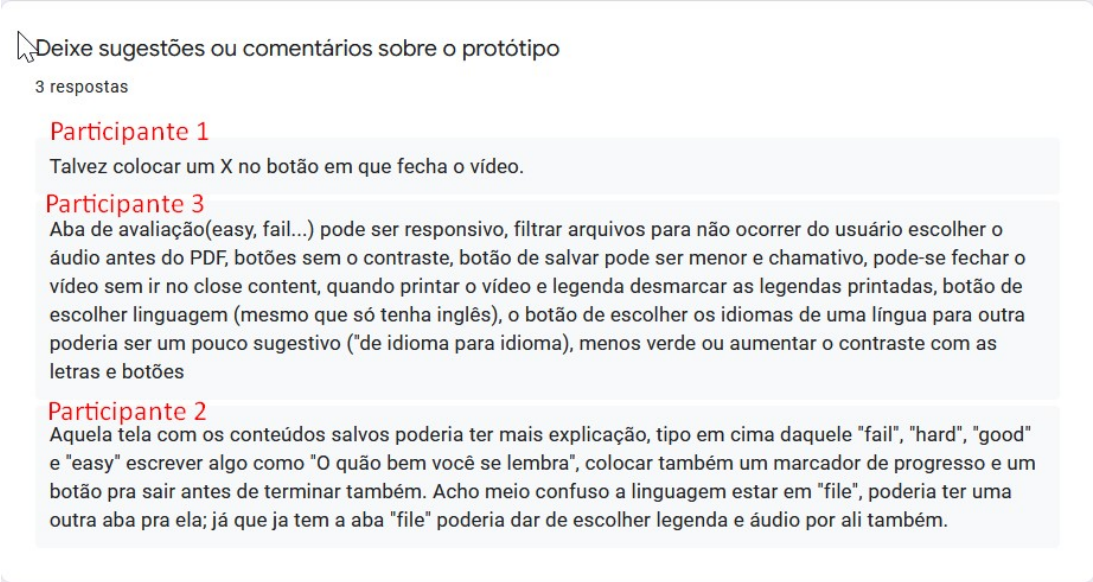
Ao final do questionário, uma caixa de texto para sugestões e comentários ficou aberta para os participantes. Tais comentários podem ser vistos na Figura 50.

Figura 49 - Participantes do teste



Fonte: Produção do autor

Figura 50 - Comentários do teste



Fonte: produção do autor

7.3 ANÁLISE DOS RESULTADOS

Para análise dos resultados obtidos no teste, é importante estar ciente de que o a amostra de participantes era pequena, e os resultados poderiam variar bastante caso a amostra fosse maior. Para termos de eficácia e eficiência, a Tabela 18 traz a comparação da expectativa de conclusão levantada como requisito e o valor real obtido. A coluna “Eficiência alcançada” traz a quantidade dos participantes que realizaram a atividade abaixo ou igual ao tempo estipulado.

Tabela 18 - Síntese dos resultados – Eficácia e eficiência

#	Atividade	Eficácia esperada	Eficácia alcançada	Eficiência	Eficiência alcançada
1	Selecionar um idioma para estudo no sistema	3/3	3/3	18 segundos	3/3
2	Carregar uma mídia de vídeo para estudar	3/3	3/3	24 segundos	3/3
3	Carregar uma legenda para o vídeo carregado	3/3	3/3	24 segundos	3/3
4	Usar o dicionário para pesquisar alguma palavra	3/3	3/3	18 segundos	3/3
5	Salvar uma entrada do dicionário em um flashcard	3/3	3/3	9 segundo	1/3

6	Salvar uma captura de tela em um flashcard	3/3	3/3	27 segundos	3/3
7	Salvar trechos da transcrição de legenda em um flashcard	3/3	3/3	27 segundos	3/3
8	Abrir um baralho e revisar um cartão	3/3	2/3	30 segundos	2/3 (uma desistência)
9	Fechar a mídia de vídeo aberta	2/3	3/3	21 segundos	1/3
10	Carregar PDF + áudio (audiobook)	3/3	3/3	58 segundos	3/3

A atividade 8 foi a única que realmente falhou em termos de eficácia, e foi devido ao fato do participante não entender que SRS, Spaced Repetition System, tratava-se de revisões, talvez a utilização de um termo menos técnico na aba de SRS, como "*Review*", poderia ter ajudado a evitar esse desentendimento. Ao contrário do esperado, ninguém falhou na atividade 9, sobre fechar uma mídia aberta.

Para eficiência, pode-se observar na atividade 5, Salvar uma entrada do dicionário em um flashcard, que os participantes tiveram dificuldades para entender como salvar uma entrada do dicionário em um flashcard, o feedback comum recebido foi que o botão parecia uma decoração, e acabou sendo ignorado. Para a atividade 9, apesar de todos terem conseguido realizá-la, apenas um deles terminou dentro do tempo estimado, o problema principal se dava ao fato da ação de fechar mídia estar um pouco escondido.

Finalmente, para satisfação, um dos participantes rendeu uma pontuação abaixo dos esperados 75 pontos, e os outros dois renderam pontuação maior ou igual. Contudo, o participante 1 rendeu uma pontuação de 100 pontos, mas foi possível observar durante todo o processo de teste que o sistema não está em "um nível de 100 pontos", então é bom considerar as respostas do SUS do participante 1 com um pouco de cautela.

De modo geral, o teste gerou um ótimo feedback para saber em que pontos o sistema pode melhorar, mas sempre importante ter em mente que a amostra não era muito grande, e que então a confiabilidade dos resultados não é de 100% (vide questionário SUS com mais de 100 pontos).

8 CONCLUSÃO E CONSIDERAÇÕES FINAIS

Neste trabalho, desenvolveu-se um protótipo de um sistema para auxílio na aprendizagem de idiomas estrangeiros, que utiliza a abordagem comum de imersão em um idioma por meio de mídias nativas e autênticas, paralelamente a fazer anotações para salvar em flashcards para revisões automatizadas e otimamente espaçadas a fim de tirar proveito do *spacing effect* (Ebbinghaus, 1885). O protótipo desenvolvido visa otimizar o tempo de estudo possibilitando criação rápida de flashcards por meio de poucos cliques de botões bem como disponibilizando um sistema de dicionário na própria plataforma, de modo a agilizar o processo de pesquisa por palavras desconhecidas. Assim, foi desenvolvido um protótipo com três módulos principais: para mídia, para dicionário e de repetição espaçada (SRS), sendo possibilitado pelo sistema um bom nível de comunicação entre eles, visando otimizar o tempo de estudo.

Desenvolveu-se, também, um extrator de dicionário que busca definições no Wiktionary, o extrator é consideravelmente genérico, podendo extrair definições de múltiplos idiomas, e apesar de ter sido desenvolvido com o domínio em inglês do Wiktionary, o bot é facilmente expandido para outros idiomas de domínio, vide APÊNDICE A – Expandindo o DSPD para outros idiomas. Os dados extraídos durante o trabalho foram utilizados no módulo de dicionário do INTLIN, contudo o DSPD é um sistema a parte e pode ser utilizado para outros fins.

8.1 LIÇÕES APRENDIDAS

No início do projeto, definiu-se um escopo demasiadamente grande para o protótipo, com muitas funcionalidades, não havendo tempo de incluí-las no protótipo inicial, assim bastante do escopo foi cortado com o desenrolar do projeto. Também, houve funcionalidades planejadas que não foram possível por limitações das tecnologias escolhidas, e.g. assim como é possível extrair imagens de vídeos para adição em flashcards, desejava-se extrair trechos de áudios para salvar, contudo a API padrão de mídias do JavaFX não permitia tal ação, existiam algumas bibliotecas para tal, contudo bastante antigas e desatualizadas, gerando bugs inesperados, cortando-se assim essa funcionalidade do escopo.

Extraiu-se opiniões valiosíssimas de membros do público-alvo durante a realização do teste. Pode-se coletar diversos feedback de onde o sistema pode melhorar, tais como uma melhor indicação das revisões, uma forma mais eficiente em termos de usabilidade para salvar entradas de dicionários em flashcards, questões quanto ao log de legenda, menu *file*, dentre outros.

8.2 POSSIBILIDADES DE TRABALHOS FUTUROS

A partir deste trabalho, existem dois caminhos que se pode seguir de trabalho, um com o DSPD e outro com o INTLIN.

Para o INTLIN, caberia implementar mais funcionalidades e prestigiar melhorias necessárias vinda do teste, de modo a transformar o protótipo em uma versão com cara de “Produto finalizado”.

Para o DSPD, é possível utilizá-lo para qualquer motivo que se deseje informações linguísticas, e.g. para processamento de linguagem natural (PLN). É possível expandir o bot para outros idiomas, adicionar ou remover informações extraídas das páginas, ou qualquer alteração necessária.

Bibliografia

KRASHEN, Stephen, The Input Hypothesis. issues and implications. Harlow: Longman, 1985.

Lightbown, Patsy M.; Nina Spada – How Languages Are Learned (Oxford Handbooks for Language Teachers). 3ª edição. Oxford: Oxford University Press, 2006.

Lyu, Dayan, A Critical Review of Krashen's Input Hypothesis: Three Major Arguments. Estados Unidos da América: American Research Institute for Policy Development em Journal of Education and Human Development, 2015.

Brown, H. D. – Principles of Language Learning and Teaching. 5ª edição. Londres: Pearson Education ESL, 2006.

McLaughlin, B, Theories of Second Language Learning. London: Edward Arnold, 1987.
Garcia, Damaris, Spaced Learning: Its Implications in the Language Classroom. Costa Rica: Revista de Lenguas Modernas, 2014.

O que é o Duolingo?, Duolingo, 2020. Disponível em: <https://support.duolingo.com/hc/pt-br/articles/204829090-O-que-%C3%A9-o-Duolingo->. Acesso em: 04/05/2022.

Grygo, Agnieszka; Gajek, Elżbieta, Risks of Using Duolingo by Polish Learners at Primary Level. Países Baixos: EnetCollect WG3 & WG5 Meeting, 2018.

Aoki, Yuta. Japanese Guy Tries Duolingo Japanese. Youtube, 20 Abr. 2022, 2022.

Yabla, Yabla, 2022. Disponível em: <https://www.yabla.com/>. Acesso em: 09/05/2022.

Gamma, Erich; Helm, Richard; Johnson, Ralph; Vlissides, John – Design Patterns: Elements of Reusable Object-Oriented Software. 1ª edição. Boston: Addison-Wesley, 1994.

Kent, Beck – Test-Driven Development By Example. Boston: Addison Wesley, 2002.

Wazlawick, Raul Sidnei – Engenharia de Software – Conceitos e Práticas. 2ª edição. Florianópolis: GEN LTC, 2019.

Sirisuriya, SCM de S, A Comparative Study on Web Scraping. Ratmalana: Proceedings of 8th International Research Conference, KDU,, 2015.

Items, Scrapy Documentation, 2008-2022. Disponível em: <https://docs.scrapy.org/en/latest/topics/items.html>. Acesso em: 25/06/2022.

Selectors, Scrapy Documentation, . Disponível em: <https://docs.scrapy.org/en/latest/topics/selectors.html>.. Acesso em: 25/06/2022.

Item Pipeline, Scrapy Documentation, 2008-2022. Disponível em: <https://docs.scrapy.org/en/latest/topics/item-pipeline.html>. Acesso em: 25/06/2022.

Brooke, John, SUS: A quick and dirty usability scale. Earley: , 1995.

AutoThrottle extension, Scrapy Documentation, 2008-2022. Disponível em: <https://docs.scrapy.org/en/latest/topics/autothrottle.html>. Acesso em: 08/03/2022.

Spiders, Scrapy Documentation, 2008-2022. Disponível em: <https://docs.scrapy.org/en/latest/topics/spiders.html>. Acesso em: 08/03/2022.

Ebbinghaus, Hermann – Memory: A Contribution to Experimental Psychology. New York: Teachers College, 1885.

GLOSSÁRIO

Flashcards	Cartão contendo informações na frente e no verso, usados para auxiliar na memorização
Repetição espaçada	Técnica de aprendizado baseada nas curvas de esquecimento, evidenciadas por Hermann[2], consiste em espaçar revisões em perídios concordantes com a dificuldade e recenticidade do conteúdo estudado, tendendo a revisar mais vezes conteúdos difíceis e recentes em comparação à fáceis e antigos.
Web Scraper	Bot que acessa um ou mais sites na internet e coleta algum tipo de dado.
Input	Em termos linguísticos, significa *escutar* e *ler*, em oposição à *output*, que seria falar e escrever.

APÊNDICE A – Expandindo o DSPD para outros idiomas

O DSPD foi inicialmente projetado usando a página em inglês do Wiktionary, o que possibilitava **apenas** a extração de dados *para* a língua inglesa, isto é português *para* inglês, francês *para* inglês, etc. Contudo, as páginas do Wiktionary tem o HTML bem consistentes entre os domínios em diversos idiomas, o que possibilita com algumas novas configurações expandir o bot para extrair para novos idiomas. Este apêndice exemplifica a expansão do bot para comportar a extração para o português, do modo mais minimalista possível, isto é, a aranha fica funcional porém ainda caberiam alguns polimentos.

O primeiro passo foi adicionar um novo objeto de configuração, similar ao visto na seção "*6.1.3.1 Configurações da aranha*". As novas configurações podem ser vistas na Figura 51.

Figura 51 - Novas configurações

```
languages_pt = {
    'en': {'language': "Inglês",
           'word_classes': ["Expressão", "Substantivo", "Verbo", "Adjetivo", "Pronome", "Artigo",
                            "Contração", "Preposição", "Provérbio", "Pronome Próprio", "Frase",
                            "Advérbio", "Conjunção", "Numeral", "Interjeição", "Símbolo", "Sufixo", "Prefix
                            "Locução substantiva"],
           'start-url-crawler': ['https://pt.wiktionary.org/wiki/Categoria:Inglês'],
           },
    'es': {'language': "Espanhol",
           'word_classes': ["Expressão", "Substantivo", "Verbo", "Adjetivo", "Pronome", "Artigo",
                            "Contração", "Preposição", "Provérbio", "Pronome Próprio", "Frase",
                            "Advérbio", "Conjunção", "Numeral", "Interjeição", "Símbolo", "Sufixo", "Prefix
                            "Locução substantiva"],
           'start-url-crawler': ['https://pt.wiktionary.org/wiki/Categoria:Espanhol'],
           },
}

lang_base_to_language = {
    'en': languages,
    'pt': languages_pt,
}
```

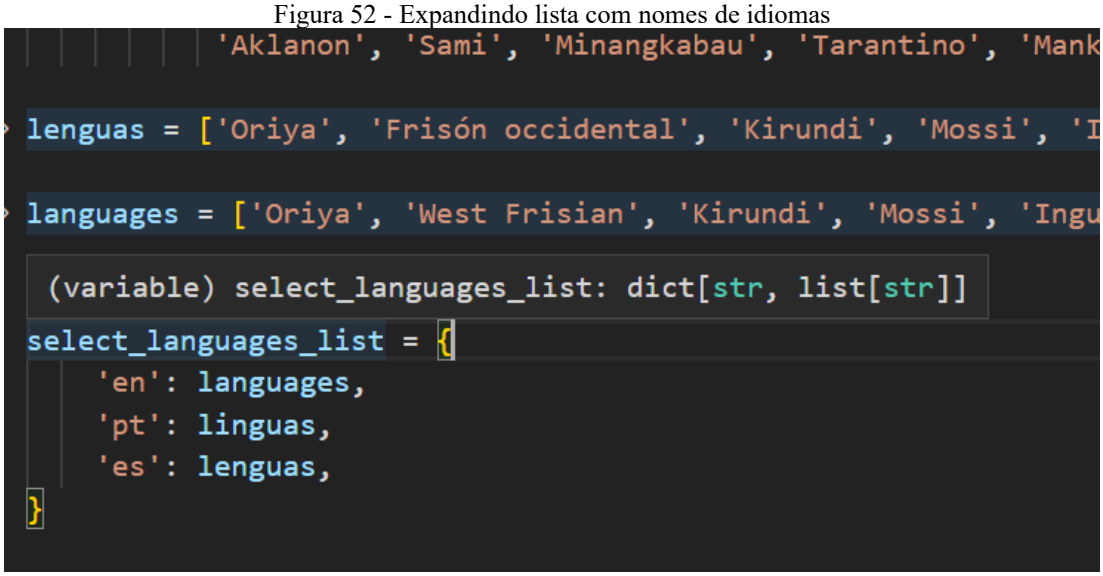
Fonte: Produção do autor

Em questão de código, essa mudança implica em alterar toda a chamada de `languages_settings.languages` para `languages_settings.lang_base_to_language[self.base_lang]` utilizando o valor base_lang, passado por linha de comando na hora de execução, para identificar qual objeto de configuração deve ser usado.

Para algumas lógicas internas, é mantido uma lista com nomes de vários idiomas, todos eles também em inglês. Assim, foi necessário uma lógica similar ao arquivo das configurações e mapeado o código ISO de dois dígitos para a lista de idiomas, como mostra a Figura 52. A lista, por ser muito extensa, foi traduzida com tradutor automático, não sendo garantido uma acurácia completa.

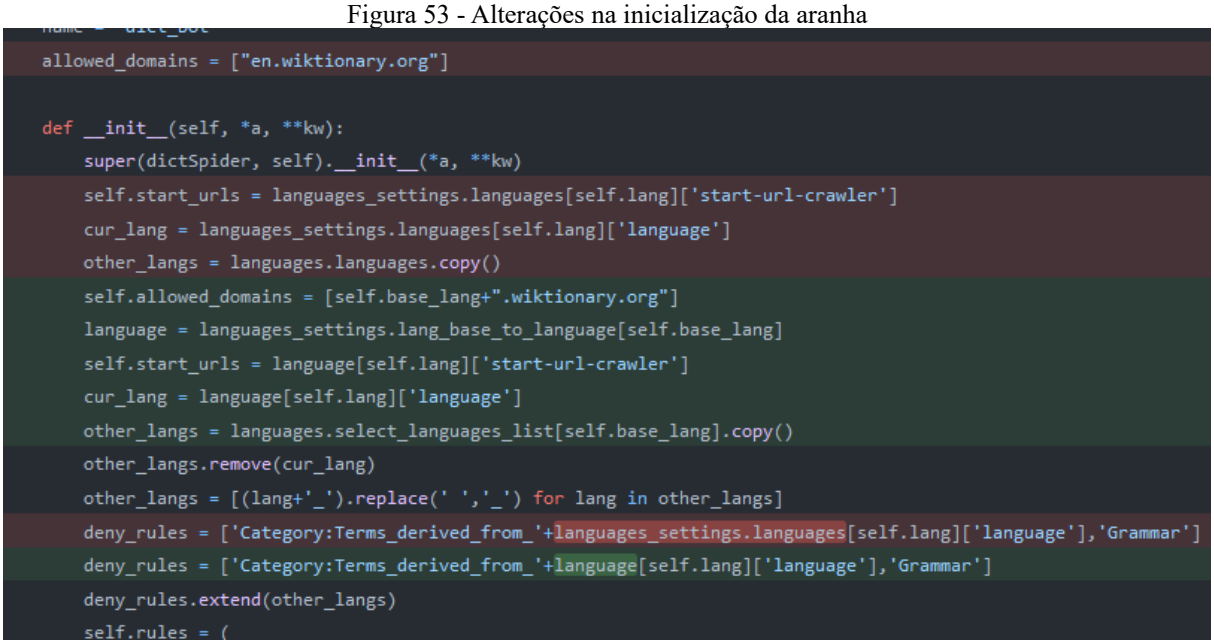
Uma última alteração foi trocar o campo allowed_domains, que estava sendo atribuído estaticamente, para uma atribuição do tipo `self.allowed_domains = [self.base_lang+".Wiktionary.org"]`.

Figura 52 - Expandindo lista com nomes de idiomas



Fonte: produção do autor

Figura 53 - Alterações na inicialização da aranha

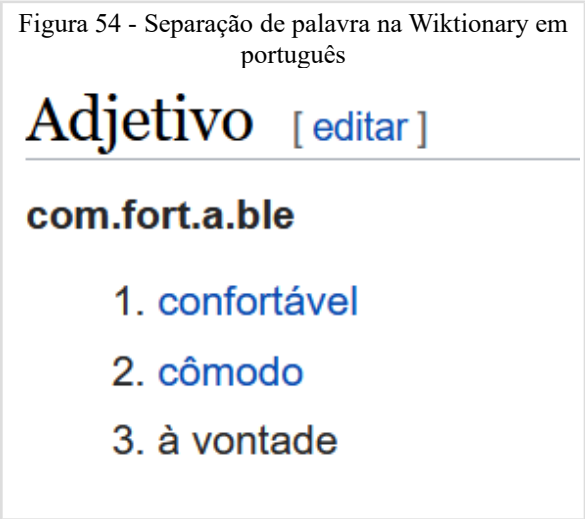


Fonte: Produção do autor

A Figura 53 acima demonstra as alterações mencionadas aplicadas no código. Com essas alterações o extrator funciona, com o mínimo de mudanças possíveis. Porém um certo nível de configuração ainda poderia ser aplicado, por exemplo a deny_rules só funciona em páginas em inglês. Poder-se-ia criar um campo de deny_rules nas configurações por idioma, de modo a criar regras de negação de urls personalizadas por base_lang.

Além disso, pode ser necessário utilização de pipelines diferentes de acordo com diferenças de páginas em outros idiomas. Na própria Wiktionary em português ocorre uma particularidade que não existe na página em inglês: algumas palavras aparecem separadas por sílabas, como pode se observar na Figura 54. Não é desejável que a palavra na saída tenha essas divisões, então foi implementado um script, que se encontra no repositório do DSPD, que faz a limpeza desses dados no banco, contudo a maneira mais elegante de se resolver isso seria a utilização de um pipeline em que se realiza esse tipo de limpeza para páginas da Wiktionary em português.

Figura 54 - Separação de palavra na Wiktionary em português



Fonte: Captura do autor[20]

20 Disponível em: <https://pt.Wiktionary.org/wiki/comfortable> Acesso em 21/06/2022

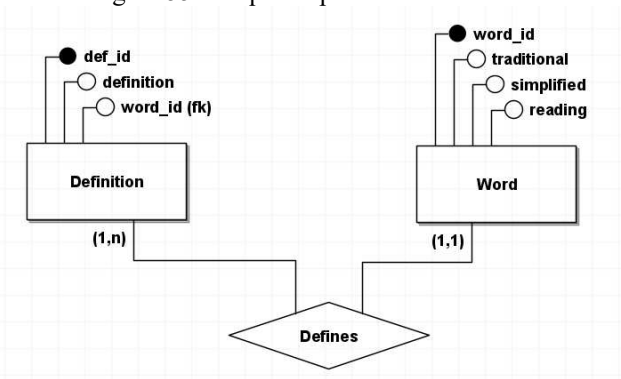
APÊNDICE B – Adicionando novos dicionários ao INTLIN

Como discutido em 6.2.1.1.1 Arquitetura do modelo dos dicionários o modelo de dicionários é desenvolvido de modo a ser genérico o suficiente para que seja possível expandi-lo com outros modelos de banco de dados de dicionários.

Assim, foram encontrados outros dois conjuntos de dados bastante completos para os idiomas Japonês e Mandarim, Jmdict e CC-Cedict, respectivamente. Ambos de licença livre dadas as devidas atribuições[2122]. O Jmdict está em um arquivo em XML e o CC-Cedict em um formato de texto próprio. Ambos foram convertidos para SQLite. Realizou-se apenas uma tradução direta para SQLite dos seus arquivos originais, dentro do possível.

A Figura 55 mostra o modelo do CC-Cedict, o arquivo era bastantes simples, gerando apenas essas duas tabelas. Já na Figura 56 demonstra o modelo chegado para o Jmdict a partir de seu XML, e como possível observar, de um grau de complexidade bem maior. As chaves estrangeiras foram omitidas para tentar diminuir a quantidade de informações na imagem.

Figura 55 - Esquema para BD CC-Cedict

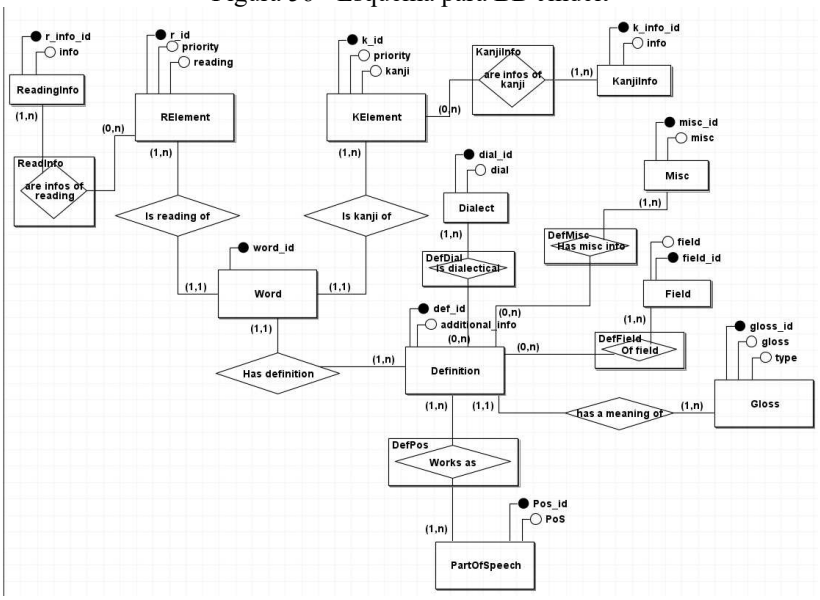


Fonte: produção do autor

21 Disponível em: <http://www.edrdg.org/wiki/index.php/JMdict-EDICT_Dictionary_Project#COPYRIGHT> Acesso em: 14/08/2021

22 Disponível em: <https://www.mdbg.net/chinese/dictionary?page=cedict> Acesso em: 14/08/2021

Figura 56 - Esquema para BD Jmdcit



Fonte: produção do autor

Assim, tomando como base a Figura 24, deve-se criar um parser que converta esses arquivos para o modelo de banco de dados definido a fim de que se possa ser usado pela classe de dicionário do sistema. Após isso deve ser implementada uma classe que estenda a classe abstrata de dicionário para cada um dos CC-Cedict e Jmdict.

Apenas meio caminho foi concluído, não tendo sido implementadas as classes de dicionário em si, apenas seus parsers. Assim como a maior parte do modelo do sistema, esses parsers estão cobertos por uma gama de testes unitários a fim de garantir uma boa cobertura do seu funcionamento. Para as classes de dicionário, que não foram implementadas, seria necessário sobrescrever os métodos abstratos e implementar métodos específicos do dicionário.

O próximo passo após ter o modelo concluído seria apresentá-los na interface gráfica. Devido à diferença nos esquemas de dicionários, haveriam algumas informações diferentes que se desejaria exibir em relação ao modelo de dados padrão do INTLIN. Seria necessário utilizar estratégias diferentes para construção dos resultados da consulta nos diferentes tipos de dicionários, para isso, o controller, que é uma classe *observable,* poderia notificar a view do tipo de dicionário carregado, fazendo a view adaptar sua estratégia de renderização de dicionário automaticamente de acordo com o tipo de dicionário carregado no sistema.

APÊNDICE C – Código fonte

Intlin: Disponível em: https://github.com/GabrielRF1/Intlin

DSPD: Disponível em: https://github.com/GabrielRF1/DSPD

APÊNDICE D – Artigo Formato SBC

INTLIN: UM SISTEMA PARA AUXÍLIO NO ESTUDO DE LÍNGUA ESTRANGEIRAS

Gabriel R. Ferreira[1]

[1]Departamento de Informática e Estatística – Universidade Federal de Santa Catarina (UFSC)

gabriel.ramildes.ferreira@grad.ufsc.br

Abstract. *A common way to learn a new language used by self-taught students is to expose oneself to the target language, via authentic media, while searching for unknown vocabulary. For the scheduling of revisions, flashcards are also used in a spaced repetition system. The biggest challenge of this method is that the learner has to handle an array of unrelated tools. Besides having their content loaded, they have to use some sort of external dictionary; as for the flashcards, besides being necessary to install some spaced repetition system, some students might want to write down their newly acquired information so that they can later create the flashcards, or even, they might have to stop the studying of their media in order to create them.*

Resumo. *Um método de aprendizado de idiomas muito comum utilizado por estudantes autodidatas é o de exposição ao idioma estudado, via consumo de mídias autênticas, concorrentemente pesquisando vocabulários, conforme a necessidade. Utiliza-se, também, flashcards em um sistema de repetição espaçadas para tratar de revisões. O maior problema desse método é que o estudante deve administrar diversas ferramentas. Além de ter seu conteúdo aberto, deve procurar algum dicionário externo; para os flashcards, além de necessário instalar algum sistema de repetição espaçada, é necessário anotar em algum lugar as informações para os cartões que se deseja criar, ou até mesmo, parar tudo que está fazendo para criá-los.*

1. Introdução

Dentro de métodos de aprendizagem de idiomas, abordagens que tem como principal princípio a exposição massiva a input, preferencialmente de conteúdos de interesse pessoal do estudante são muito populares entre estudantes. A teoria de aprendizagem via input foi levantada na acadêmia pela primeira vez pelo Professor da Universidade do Sul da Califórnia, Stephen Krashen (Kreshen, 1985). Apesar de bastante difundida e ter gerado uma mudança de paradigmas de como idiomas são ensinados em sala de aula (Patsy & Spada, 2006) (Dayan, 2015), sua hipótese do input (Input Hypothesis, no original) é uma das hipóteses de aquisição de segunda língua mais controversas (Brown, 2006) no meio acadêmico. Dayan Liu afirma que a hipótese de Krashen é vaga, não havendo definições exatas de seus conceitos (Dayan, 2015), outra crítica seria que as suas teses seriam impossíveis de um teste empírico (McLaughlin, 1987). Mesmo com tais objeções, o trabalho de Krashen é muito influente e de bastante importância em pesquisas da linguística, tendo surgido adaptações de sua hipótese que tentam suprir eventuais deficiências teóricas (Dayan, 2015), bem como diversas pesquisas e estudos em cima de sua hipótese. Além de métodos de aprendizagem baseados em input terem sido amplamente difundidos, há pesquisas indicando que estudantes, que já tenham um

certo nível, conseguem um progresso considerável por exposição a uma língua alvo, sendo recomendado como um método suplementar para estudantes intermediários ou avançados (Patsy & Spada, 2006).

Como mencionado, a hipótese de Krashen foi muito difundida e desde de quando foi proposta, surgiu e adaptaram-se diversos métodos seguindo essa linha como base. Dentre internautas, é muito comum o sentimento que exposição massiva ao idioma por meio de input com conteúdos de interesse pessoal seria a forma mais eficiente de aprender tal língua. Acima disso, frequentemente são usados sistemas de repetição espaçada, em que o estudante cria flashcards (seja de vocabulário, sentenças, diálogos, ou qualquer outro a gosto) que são cartões que serão apresentados para revisão em períodos de tempos variantes, feito de modo a tomar proveito do spacing effect, fenômeno psicológico observado pela primeira vez por Hermann Ebbinghaus (1885)[2]. Tais cartões possuem uma frente e um verso, a frente possui o conteúdo principal a ser estudado, e o verso possui informações para que o estudante verifique se realmente acertou o conteúdo do cartão, ou o relembre em caso de erro. Há diversos benefícios da repetição espaçada no aprendizado, dentre outros, ela aprimora a habilidade de lembrança do estudante (Damaris, 2014).

Deste modo, um plano de estudo baseado no especificado anteriormente, usualmente segue o seguinte fluxo descrito na Figura 1. (1) O estudante escolhe uma mídia de seu interesse e em nível apropriado, (2) ele consome dada mídia, atento a estruturas e vocabulários não conhecidos, (3) realiza uma pesquisa sobre os pontos desconhecidos encontrados e os anota (4) cria-se flashcards para revisão deste novo conteúdo. (5) O estudante realiza a revisão de cartões agendados para o dia.

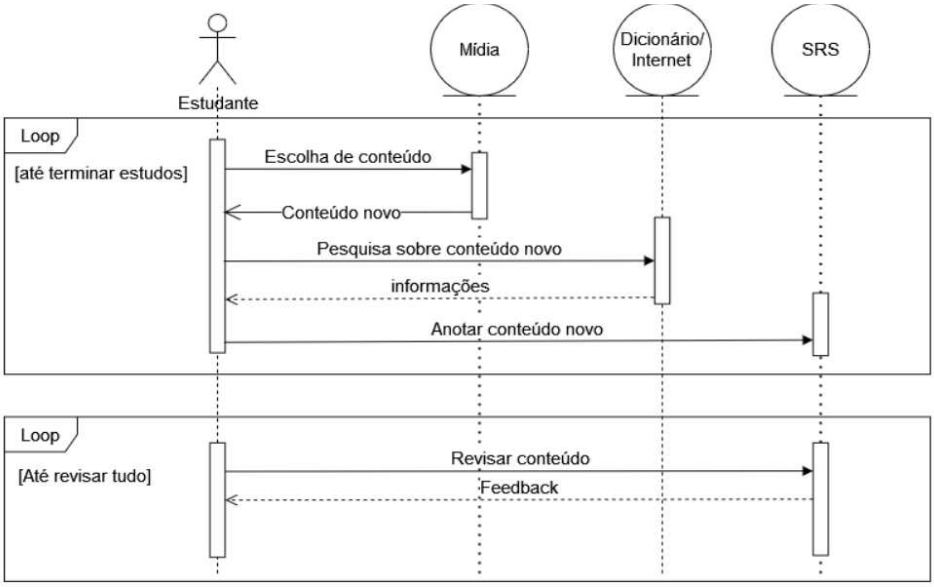


Figura 1. Diagrama de sequência do fluxo de estudos

1.1. Problemática

Considerando o plano de estudos especificado na seção anterior, um estudante desejará utilizar algum recurso para tal modo de estudo, contudo a escassez de um sistema único que agrupe, gratuitamente, todos os recursos necessários para o seguimento de tal plano

de estudo se torna evidente.

Por meio da internet, o trabalho de encontrar alguma mídia é trivial. O acesso a informações sobre vocabulários e estruturas do idioma também são amplamente acessíveis. Outrossim, existem diversos softwares de repetição espaçadas gratuitos que podem ser utilizados para o fim de anotações e revisões, sendo o Anki um dos mais populares. Desta forma, apenas com recursos gratuitos já seria possível realizar estudos deste tipo, porém tais recursos são todos desconexos entre si, necessitando tanto da administração de múltiplos recursos, como que haja uma pausa do consumo da mídia em estudo para realização de consultas e anotações, é possível imaginar como a constante remoção de foco do conteúdo estudado pode tornar o processo de estudo pouco fatigoso e menos engajante. Ademais, sistemas de repetição espaçadas (ou SRS) costumam permitir inserção de mídias nos cartões, servindo como um complemento áudio ou visual aos textos nos cartões ou, até mesmo como o conteúdo em si do cartão, isso exige do estudante realizar capturas de telas manualmente, cortar clipes de áudio ou vídeo com algum outro recurso a parte, para que então possa inserir esse pedaço de mídia em seu cartão.

Assim, o problema principal se dá pela dificuldade de encontrar um sistema que integre diversos recursos em um único sistema, para que o processo de estudo seja mais fluído e prazeroso.

1.2. Objetivos

Este trabalho propõe desenvolver um sistema para auxiliar o estudo de idiomas estrangeiros por método baseado em input, bem como um web scraper para coleta de dados de dicionários, gerando como subproduto um grande conjunto de dados de dicionários, que ficará disponível em domínio público.

A plataforma desktop para auxiliar no aprendizado de idiomas estrangeiros integra em um único pacote: (i) um sistema de dicionários de múltiplos idiomas, (ii) um sistema de repetição espaçada próprio e (iii) um módulo de recursos multimídia, em qual é possível abrir mídias para estudo a gosto. Visando, assim, agilizar o processo de criação de flashcards e manter todos os recursos em um único pacote para que seja possível manter o foco na mídia principal estudada. Tendo como especificações ser um sistema para uso gratuito, que permita a criação de flashcards tanto manualmente como automaticamente a partir de recursos do sistema, possua um sistema interno de dicionário, bem como permitir que o usuário abra mídias de conteúdos de interesse próprio. Tal sistema será nomeado de INTLIN, a junção de integrated (integrado, do inglês) com língua, do português.

Para o sistema de dicionários, deseja-se montar um modulo completamente offline, para isso será necessário conseguir um conjunto de dados que possam ser utilizados pelo sistema, sem necessidade de consumir uma API online. Deste modo, além do principal aplicativo proposto, desenvolver-se-á um bot em para extrair dados de dicionário, a partir do site Wiktionary. Esse robô será desenvolvido em Python utilizando o framework para web scraping, Scrapy. Esse web scraper será nomeado de DSPD, acrônimo para Dictionary Spider.

2. Trabalhos relacionados

No mercado, existem sistemas similares ao descrito, mas não abrangendo todas as

especificações colocadas na seção anterior: sistema para uso gratuito, criação de flashcards (manualmente e automaticamente), sistema interno de dicionário e possibilitar abrir mídias de conteúdos de interesse próprio para estudo. Sendo o fato de outros sistemas não serem completamente gratuitos o maior problema comum identificado.

Existem também diversos outros aplicativos gratuitos de bastante sucesso no mercado, tal como Duolingo, entretanto a proposta desse aplicativo é fundamentalmente diferente do proposto pelo INTLIN.

Os sistemas escolhidos para comparação são apenas aqueles que possuem a abordagem de estudo especificado: Utilizar mídias autênticas de interesse do estudante e ter um sistema de revisões baseados em repetição espaçada. Por isso, o sistema Duolingo ficará de fora da comparação realizada em 2.5 Comparativo geral. Mesmo assim, o mesmo recebeu uma seção própria devido a sua popularidade, bem como para esclarecer o porquê de aplicativos desse estilo serem diferentes da proposta do INTLIN.

2.1. Duolingo

Duolingo é um sistema de estudos de idioma de grande sucesso, sendo possivelmente o aplicativo mais popular de aprendizado de idiomas, tendo mais de 300 milhões de usuários, de acordo com o portal oficial (Duolingo).

Apesar de seu enorme sucesso, sua proposta é bastante diferente do que se deseja alcançar com o INTLIN. O Duolingo segue um sistema de lições ordenadas por tópicos que vão sendo desbloqueados progressivamente, similar a um sistema de níveis em um videogame. As lições são aglomerados de exercícios constituídos primariamente de tradução de sentenças, tais frases começam simples e progressivamente aumentam em dificuldade. O principal a se observar é que as sentenças dos exercícios são acompanhadas de pouco contexto, que na sua ausência pode acarretar na perda de entendimento de certas nuanças, ou até mesmo ambiguidade (Agnieszka & Elżbieta, 2018) (Yuta, 2022).

Com isso, apesar de todo o sucesso do aplicativo, o Duolingo tem uma proposta diferente do INTLIN e dos demais trabalhos a serem tratados neste capítulo, os quais têm como foco principal o estudo da língua por meio de mídias autênticas e naturalmente munidas de contexto, em oposição ao sistema de exercícios progressivos do Duolingo.

2.2. FluentU

O FluentU é uma plataforma para aprendizado de línguas, que disponibiliza uma extensa biblioteca de vídeos em diferentes níveis de dificuldade, sendo possível pesquisar definições de palavras vistas nas lições, flashcards são criados automaticamente a partir de lições estudadas, também sendo possível criá-los manualmente, conforme o estudante necessitar. Apesar de permitir criar flashcards de modo manual, é possível apenas criar sobre vocabulários, não dando muito liberdade ao estudante para criar os cartões que desejar. Outrossim, não é possível estudar qualquer mídia de interesse do estudante, ficando atado apenas a seleção de vídeos do serviço.

Assim, esta plataforma contempla as principais especificações do INTLIN, além de possuir vídeos autorais para estudo. Contudo, por não ser gratuita, não é uma ferramenta muito acessível. Desta forma, sua diferença com o INTLIN torna-se evidente. Enquanto o FluentU já é um serviço bastante robusto, ele não permite o

mesmo grau de liberdade na escolha de conteúdos e na criação de flashcards que o INTLIN ambiciona, além de ser um serviço pago.

2.3. Yabla

Este serviço é bastante similar ao FluentU, possuindo uma biblioteca extensa de vídeos e um sistema de repetição espaçada para revisão do conteúdo. Também é interessante ressaltar que o sistema é bem avaliado por diversos órgãos públicos de renome, tais como universidade de Michigan e departamento de estado dos Estados Unidos (Yabla).

Comparando-se ao FluentU, esse serviço não permite criação de flashcards manualmente, e possui uma gama menor de idiomas disponíveis. Apesar de ainda ser pago, tem a vantagem de ser mais barato. Também não é possível importar nenhum tipo de mídia própria para o sistema, estando o estudante limitado a biblioteca nativa do sistema.

2.4. LingQ

LingQ, pronunciado como “link”, é um serviço muito próximo do proposto pelo INTLIN, com diversos extras e algumas desvantagens. Neste serviço, existem diversas lições, em formato de texto, áudio e/ou vídeo, disponibilizadas pela pela plataforma. Naturalmente, é possível pesquisar vocabulários enquanto se estuda uma lição, e então salvá-los para revisão em um sistema de SRS, e para palavras salvas manualmente, flashcards adicionais com elas utilizadas em frases são criados automaticamente. Além das lições do sistema, é possível importar algum conteúdo de interesse próprio, podendo ser textos, áudios ou vídeos, de acordo com o gosto do estudante.

Além do mencionado, o LingQ vai muito além, possuindo recursos de comunidade para comunicação entre estudantes, como fóruns e intercambio de escritas. Também disponibiliza sistema para agendamento de tutorias com professores que utilizam a plataforma. Assim, o LingQ vai além de um sistema que ajuda o estudante a evoluir a compreensão de um idioma, e disponibiliza recursos para o estudante melhorar seu output, i.e. escrita e fala.

Apesar de um ótimo serviço, o uso do LingQ possui uma assinatura Premium, que gera limitações para usuários gratuitos, havendo um limite para quantidade de flashcards que podem ser criados e de lições importadas, deixando o usuário preso apenas com lições do sistema e com números limitados de flashcards, após exceder os limites.

2.5. Comparativo geral

As principais plataformas similares ao INTLIN foram discutidas nas seções anteriores. A Tabela 1, a seguir, faz uma comparação do que se propõe desenvolver com estas plataformas, de acordo com os pontos propostos para o trabalho.

X	FluentU	Yabla	Lingq	INTLIN
Sistema de dicionário interno	✔	✔	✔	✔
Flashcards automáticos e manuais	✔	✘	✔	✔
Abrir conteúdo de interesse próprio	✘	✘	✔	✔
100% Gratuito	✘	✘	✘	✔
Conteúdos autorais para estudar	✔	✔	✔	✘

Tabela 1. Comparação deste trabalho X sistemas similares

Os pontos em amarelo na comparação indicam que a plataforma contempla a funcionalidade, mas com algumas ressalvas. O FluentU não permite uma flexibilidade muito grande na criação dos seus flashcards, como já discutido em sua seção. O LingQ também só permite criar flashcards de vocabulários vistos nas lições, não permitindo que o usuário personalize seus cartões de acordo com suas necessidades.

Além disso, as outras três plataformas da comparação disponibilizam conteúdos próprios em diferentes níveis, o que não é possível oferecer no INTLIN, devido ao fato de ser uma aplicação gratuita e não haver verba para elaboração profissional de conteúdos autorais, sendo delegado ao usuário encontrar algum conteúdo apropriado de seu interesse para estudo.

Dessarte, o INTLIN não é uma plataforma que ensina um idioma, ele não possui instruções gramaticais ou explicações sobre nenhuma língua suportada, sendo seu uso recomendado como uma ferramenta auxiliar na qual o estudante pode expandir seu vocabulário e entendimento de uma língua por meio de consumo de conteúdos autênticos no idioma alvo, por conseguinte, seu público-alvo é o estudante em nível intermediário, ou maior, já com algum conhecimento básico da língua estudada.

3. Desenvolvimento

Esse capítulo detalha o desenvolvimento de todo o projeto. Primeiramente, entrando em detalhes sobre a aranha, seus comportamentos, dados a serem extraídos e seus detalhes de implementação. Na sequência, detalhar-se-á o desenvolvimento do INTLIN, tratando sobre cada módulo individualmente, bem como suas características e particularidades de desenvolvimento.

É importante ressaltar que o DSPD não é uma parte do INTLIN, ele é um sistema que rodará previamente para coletar dados que, posteriormente, serão utilizados pelo outro sistema. Assim, ele não funcionará como uma espécie de backend para o INTLIN, ele é um sistema desacoplado que também poderia ser utilizado para outros fins além da coleta de dados para o INTLIN.

Para efeitos deste trabalho, o DSPD foi utilizado para gerar dados para o modulo de dicionário do INTLIN. Ele gera vários dados, que são adicionados em um banco local SQLite, que pode ser posteriormente transferido para o INTLIN, e então usado, de mesmo modo, localmente pelo protótipo. A Figura 2 ilustra o processo.

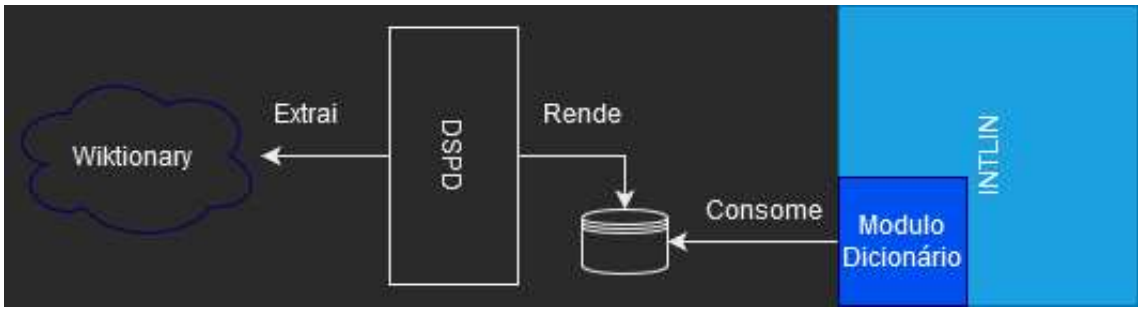


Figura 2. DSPD X INTLIN

3.1. Dictionary Spider (DSPD)

O primeiro passo, antes de construir o INTLIN, é desenvolver um extrator de dados para montar um dicionário a ser usado pelo sistema. Esta seção tratará do desenvolvimento do DSPD, iniciando com modelagem e definições e seguindo para especificações da implementação. Esta seção contemplará aspectos de comportamentos do DSPD, dados extraídos e detalhes da sua impas páginas.

O DSPD, Dictonary Spider, é um extrator desenvolvido em Python com o uso do framework Scrapy. Seu fluxo de funcionamento básico, explicitado pela Figura 3, começa com a aranha requisitando, ao Wiktionary, endereços iniciais de algum idioma específico. A partir dos links iniciais, a aranha segue para múltiplas páginas, em que realizará o parsing dos itens, e então os enviará para os dois Pipelines. O primeiro pipeline trata da remoção de dados inválidos e sujeiras que ocorrem em algumas páginas do Wiktionary, o segundo tem a função de inserir os itens coletados em um SQLite, bem como adicionar o item a um JSON de saída da execução.

Para evitar sobrecarregamento do servidor do Wiktionary, a aranha é bem conservadora, fazendo requisições lentas ao servidor, não requisitando nada em menos de 4 segundos, também utiliza um middleware de autothrottle, que regula a velocidade de requisição de acordo com o tempo de resposta do servidor (AutoThrottle, Scrapy). Acima disso, ela também obedece o arquivo robots.txt, a fim de ser respeitosa com o site e evitar banimento, visto que tal arquivo explicita que robôs não devem adentrar em certas páginas.

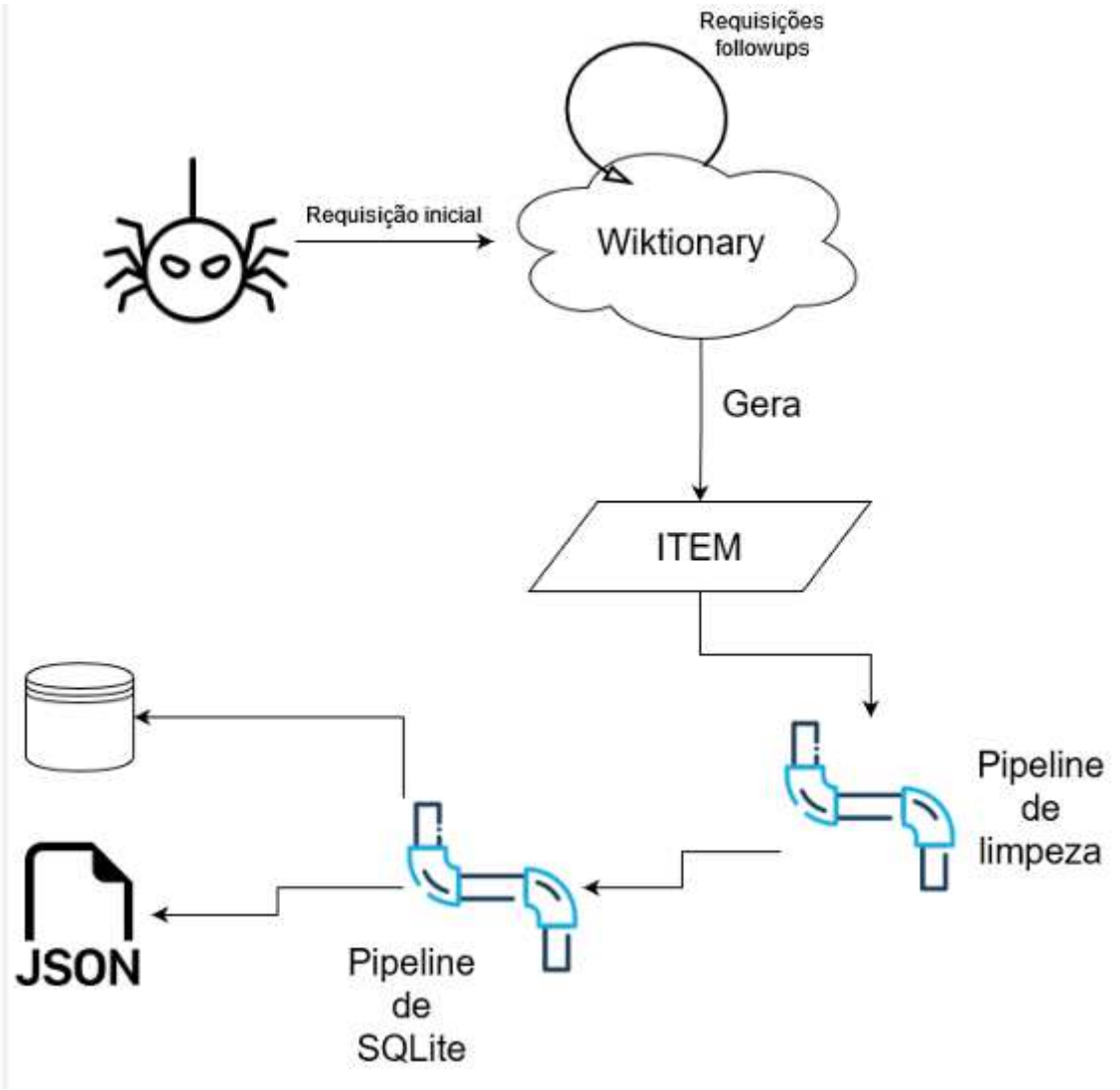
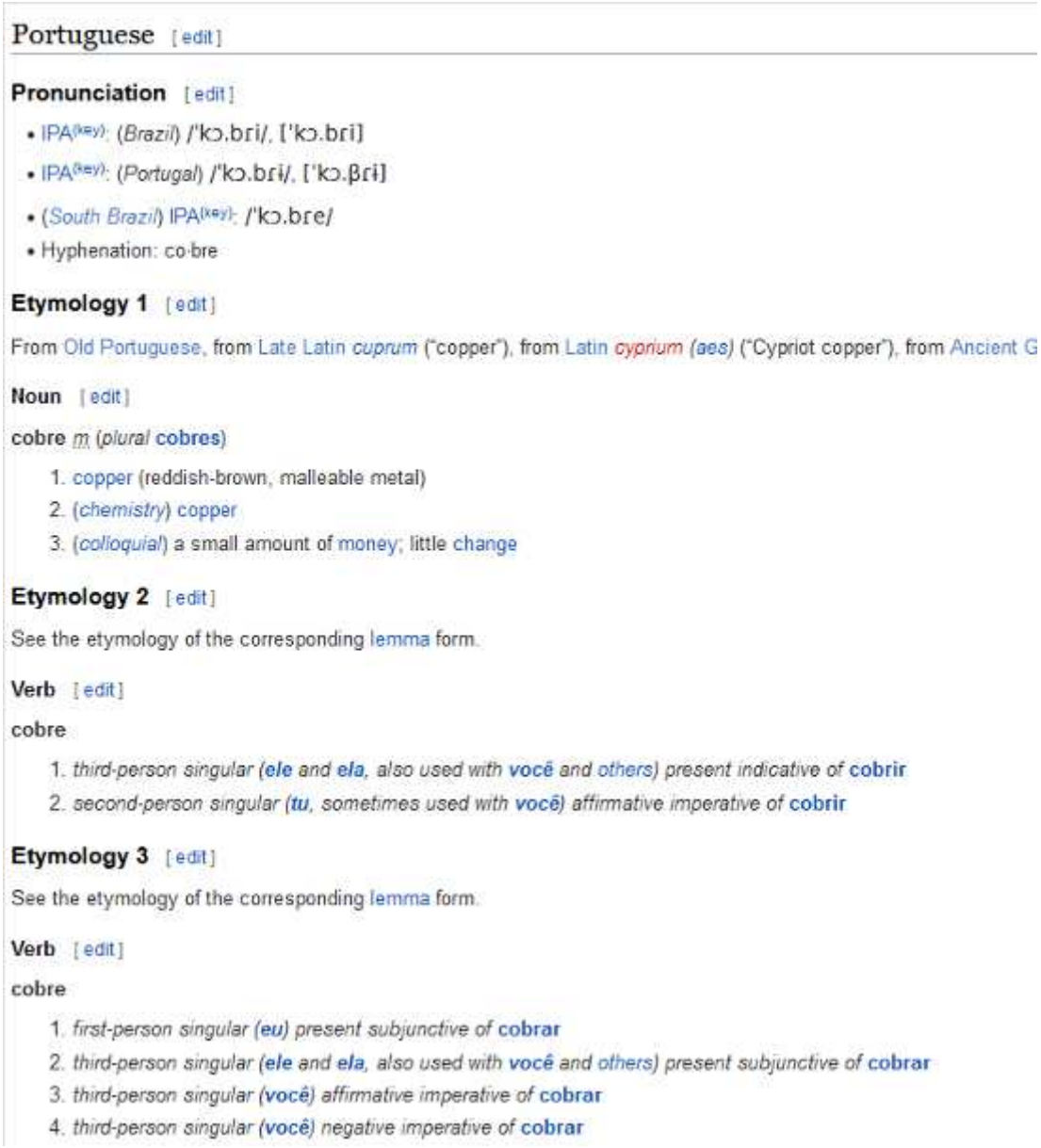


Figura 3. Fluxo geral de funcionamento do DSPD

As definições de palavras foram extraídas de páginas do Wiktionary e colocadas em um item que tem a forma de um elemento de arquivo JSON, assim posteriormente, um JSON poderá ser gerado com todos os itens. Também é realizado um parsing sobre tais itens para inseri-los em um SQLite.

Primeiro, realizou-se uma análise em que tipos de informações estão presentes em uma página do Wiktionary, bem como definir o que se deve extrair. A Figura 4, a seguir, possui um exemplo de uma página do Wiktionary e, pode-se observar que para uma mesma string, é possível que ocorram múltiplas entradas no dicionário, delimitado pelas classes gramaticais. São, de fato, entradas diferentes, o cobre substantivo não deve ser considerado a mesma entrada que o verbo cobrir ou cobrar.

Similarmente, dentro de uma entrada é possível que ocorram múltiplas definições, vide como cobre substantivo possui três definições.



Portuguese [edit]

Pronunciation [edit]

- IPA(key): (*Brazil*) /ˈkɔ.bɾi/, [ˈkɔ.bɾi]
- IPA(key): (*Portugal*) /ˈkɔ.bɾi/, [ˈkɔ.βɾi]
- (*South Brazil*) IPA(key): /ˈkɔ.bɾe/
- Hyphenation: co·bre

Etymology 1 [edit]

From Old Portuguese, from Late Latin *cuprum* ("copper"), from Latin *cyprium (aes)* ("Cypriot copper"), from Ancient G

Noun [edit]

cobre *m* (*plural* **cobres**)

1. copper (reddish-brown, malleable metal)
2. (*chemistry*) copper
3. (*colloquial*) a small amount of money; little change

Etymology 2 [edit]

See the etymology of the corresponding lemma form.

Verb [edit]

cobre

1. *third-person singular (**ele** and **ela**, also used with **você** and others) present indicative of* **cobrir**
2. *second-person singular (**tu**, sometimes used with **você**) affirmative imperative of* **cobrir**

Etymology 3 [edit]

See the etymology of the corresponding lemma form.

Verb [edit]

cobre

1. *first-person singular (**eu**) present subjunctive of* **cobrar**
2. *third-person singular (**ele** and **ela**, also used with **você** and others) present subjunctive of* **cobrar**
3. *third-person singular (**você**) affirmative imperative of* **cobrar**
4. *third-person singular (**você**) negative imperative of* **cobrar**

Figura 4. Exemplo de página do Wiktionary com múltiplas entradas

Outrossim, definições específicas de uma entrada podem possuir atributos próprios, tais como sinônimos e antônimos, a Figura 5 evidencia esse caso para o verbo dar, do português. E, de fato, essas informações pertencem as definições e não a entrada do verbo dar. Como exemplo, o sinônimo da definição 5 não se encaixa na definição número 1, e vice-versa. Adicionalmente, na mesma figura, próximo aos sinônimos e antônimos, é possível perceber alguns textos sem marcadores, nesta figura são somente exemplos, contudo ocasionalmente aparecem algumas outras informações não tão relevantes. Assim, para esse extrator, optou-se por extrair qualquer informação de uma definição, fora sinônimos e antônimos, como sendo "extras" daquela definição.

Verb [edit]

dar (*first-person singular present indicative* **dou**, *past participle* **dado**)

1. (*ditransitive*) to give
    1. (*with* ***a*** *ou* ***para*** *or an indirect objective pronoun*)
        1. to transfer one's possession of something to someone without anything in return

            Dar-te-ei *um livro. / Te* ***darei*** *um livro.*

            I **will give** you a book.

            Synonym: ceder

            Antonym: receber

        2. to hand over (to pass something into someone's hand)

            Dá*-me tua mão. / Me* ***dá*** *sua mão.*

            Give me your hand.

            Synonyms: entregar, passar

        3. to make a present or gift of

            Dei *flores à minha mulher.*

            I gave my wife flowers.

            Synonym: presentear

            Antonyms: ganhar, receber

        4. to provide a service

            A Igreja ***dá*** *conforto aos pobres.*

            The Church **gives** the poor comfort.

            Ele ***dá*** *aulas de latim.*

            He **gives** Latin classes.

            Synonym: oferecer

        5. to administer (to cause to take (medicine))

            Demo*-lo insulina. /* ***Demos*** *insulina a ele.*

            We gave him insulin.

            Synonym: administrar

        6. (*transitive*) to give; to issue; to emit

            João nos ***dará*** *recomendações.*

            John **will give** us recommendations.

            Ele gosta de ***dar*** *ordens.*

Figura 5. Exemplo de página do Wiktionary com sinônimos

Por último, algumas páginas apresentam uma forma alternativa. Analisando a Figura 6, tem-se a forma alternativa de equipe como sendo equipa. Na página do Wiktionary, essa forma alternativa não aparece agrupada com classe gramatical, sendo declarada como forma alternativa global para todas as entradas de string “equipe”. Isso, na verdade, é uma imperfeição no modelo do Wiktionary, visto que, nessa mesma página é possível encontrar equipe como o verbo equipar conjugado, o que evidentemente não deveria ter a forma alternativa “equipa”. Infelizmente, não há informações na página que indiquem para a aranha à que definição a forma alternativa

deveria ser inclusa. Assim, optou-se por respeitar o modelo do Wiktionary e adicionar a forma alternativa para cada string que aparecem sob essa forma alternativa.

Figura 6. Exemplo de página do Wiktionary com forma alternativa

Desta forma, ao entrar em uma página os dados a serem extraídos são: (i) A palavra. (ii) Forma alternativas. (iii) Classe gramatical. (iv) Definições. (v) Sinônimos, Antônimos e Extras. (vi) Não mencionado, mas também é extraído o gênero da palavra, quando houver. As demais informações presentes nas páginas, como etimologia e pronuncia, foram julgadas desnecessárias por hora e não serão extraídas.

Com os dados a serem extraídos definidos, é possível mostrar como o JSON dos dados se comporta. A Figura 7 traz um comparativo side-by-side de uma página do Wiktionary e itens do JSON de saída gerado pela aranha.

Figura 7. Comparativo saída x página - base

Desta forma, pode-se chegar ao seguinte esquema para os arquivos JSON de saída. haveriam campos para palavra (String, obrigatório), gênero (String, opcional),

formas alternativas (Lista de strings, opcional), classe gramatical (String, obrigatório), definições (Lista de JSON, obrigatório). Cada definição, por sua vez, tem como valor um JSON próprio, contendo campos para uma definição (String, obrigatório), sinônimos (String, opcional), antônimos (String, opcional) e extras (String, opcional). O trecho de código a seguir contempla o JSON, sendo o símbolo de interrogação no tipo de dados um denotador de optativo.

```
{
    word: String,
    gender: String?,
    alt: Lista<String>?,
    defs: Lista<{
            _def: String,
            antonyms: Lista<String>?,
            synonyms: Lista<String>?,
            extras: Lista<String>?,
        }>
    word_class: String,
}
```

Este modelo de dados é bastante genérico e pode comportar definições de diversos idiomas. Alguns campos, como gender, podem ser vazios, o que permite que se extraia tanto definições de idiomas com gênero, como sem gênero. Contudo, ele não é adequado para todos os idiomas, pois alguns podem necessitar de alguns campos não inclusos, o que levaria a uma necessidade de estender esse modelo, por exemplo um idioma como mandariam poderia querer um campo para pinyin (leitura da palavra).

Além desta saída em JSON, os dados também são inseridos em um SQLite, que teve seu modelo definido a partir do JSON. Como já mencionado o bot possui um pipeline que insere os itens extraídos em um banco de dados. Um SQLite torna mais eficiente para a aplicação realizar consultas, comparativamente a consultar diretamente do JSON.

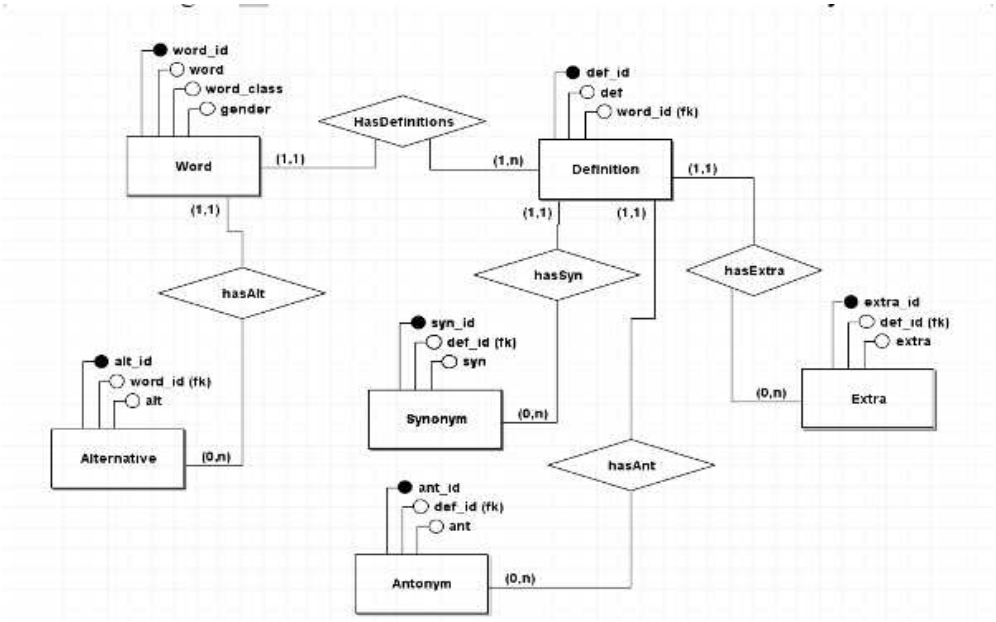


Figura 8. Modelo base de dados extraído do Wiktionary

Foram definidos uma série de valores a serem usado pela aranha dependente a qual idioma está se extraindo os dados, tais valores podem ser visualizados na Figura 9. Cada língua é identificada unicamente pelo código de 2 dígitos do padrão ISO 639-1.

O atributo language é usado para encontrar o header que inicia as definições do idioma desejado dentro do Wiktionary. O atributo de word_classes serve para a aranha encontrar as entradas específicas dentro da região do idioma, esse atributo existe pois não há diferença entre os cabeçalho de definições e outros cabeçalhos, e não se deseja, por exemplo, coletar a etimologia de uma palavra como sendo uma definição de classe gramatical “Etymology”. Por isso as classes gramáticas são definidas manualmente. Essa lista de possíveis classes não são um valor global, mas de fato um atributo interno do idioma, isso se dá pelo fato de alguns idiomas poderem ter classes gramaticais diferentes, por exemplo, uma língua como japonês não possui artigos e possuiria uma classe de “partículas” não existente em muitas línguas.

O último atributo é o start-url-crawler, esse é o ponto de partida da aranha. A escolha foi por uma página geral de categoria dos idiomas, pois a escolha de seguir links se dá de forma automática pela aranha, sendo necessário apenas um ponto de partida que leve a várias páginas do idioma.

Figura 9. Valores de configuração aranha

Além destas configurações definidas manualmente, o Scrapy traz uma série de configurações em um arquivo de *settings.py*, de modo geral as configurações selecionadas nesse arquivo, para este projeto, foram opções que tornam a aranha mais conservadora, visando não sobrecarregar o servidor do Wiktionary. Foram configuradas as opções de fazer a aranha obedecer o arquivo *robots.txt*, o qual impede a aranha de entrar em páginas que não são permitidas para bots. Foi configurado o tempo mínimo de delay entre requisições como 4 segundos, o que deixa a aranha bem lenta, mas garante que não gerará distúrbios no servidor. Também foi habilitado o mecanismo de autothrottle, que é uma funcionalidade que serve para regular a velocidade de requisições de acordo com o tempo de resposta do servidor, i.e., caso o Wiktionary demore para responder uma requisição, a aranha passará a mandar requisições mais lentas, até o tempo de resposta do servidor estabilizar. A Figura 10, a seguir, demonstra as configurações do autothrottle.

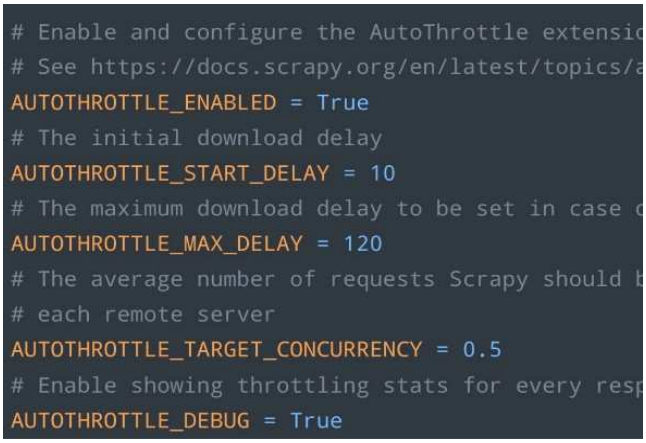


Figura 10. Configurações de autothrottle

Também é declarada uma lista de domínios permitidos, sendo aqui apenas o Wiktionary (en.wiktionary.org/) permitido, pois não se deseja que a aranha saia internet afora em direção outros sites.

O framework do Scrapy disponibiliza diversos tipos de aranhas, uma delas é a CrawlSpider, neste modo de aranha, são definidos um conjunto de regras para extração de links e seguimento dos mesmos (Spiders, Scrapy), este é o tipo de aranha usada atualmente pelo DSPD. Nos primeiros esboços do DSPD era utilizada a aranha padrão, no qual o seguimento dos links eram definidos via código, contudo, ela se mostrou ineficiente, não gerando uma quantidade boa dados. Então, optou-se pela CrawlSpider, com ela, necessita-se apenas de um link inicial qualquer e um conjunto de regras bem definidas para a que a aranha adentre em todas as páginas que encontrar de um idioma especificado. Também vale ressaltar que um link uma vez visitado, não será acessado novamente.

As regras definidas podem ser vistas na Figura 11. É possível observar na Figura abaixo duas regras sendo definidas, ambas recebem um extrator de links, esse que é responsável por encontrar links que deverão ser seguidos pelo bot. Tais extratores estão usando seletores css, para selecionar partes específicas da página, ou seja, eles coletarão os links em qualquer tag capturada pelos seletores especificados.

```
super(dictSpider, self).__init__(*a, **kw)
self.start_urls = languages_settings.languages[self.lang]['start-url-crawler']
cur_lang = languages_settings.languages[self.lang]['language']
other_langs = languages.languages.copy()
other_langs.remove(cur_lang)
other_langs = [(lang+'_').replace(' ','_') for lang in other_langs]
deny_rules = ['Category:Terms_derived_from_'+languages_settings.languages[self.lang]['language'],'Grammar']
deny_rules.extend(other_langs)
self.rules = (
Rule(link_extractor=LinkExtractor(
    restrict_css=[
        'div.CategoryTreeItem',
        'div.mw-parser-output > ul a[title^=\'Category:\']'
    ],
    deny=deny_rules
    )
),
Rule(link_extractor=LinkExtractor(
    restrict_xpaths=[
    '//*[@lang = \''+self.lang+'\' and (not(ancestor::*[@class=\'p-form-of\']) and not(ancestor::*[@class=\'form-of\']) and not(ancestor::*[@class=\'NavFrame\']))]'
    ],
    restrict_css=[
        'div.mw-category-group',
        'td#oldest-pages',
        'td#recent-additions']), callback='parse_word_page', follow=True),
)
super(dictSpider, self)._compile_rules()
```

Figura 11. Regras de seguimento de links do DSPD

A primeira regra serve para páginas de tópicos e categorias, tais páginas possuem links para outros endereços que não se quer realizar o parsing, mas ainda se deseja seguir para seus links. Nestas páginas há subcategorias, que não são páginas a se extrair dados, mas é importante segui-la, pois eventualmente serão alcançadas páginas para extração. Nesta primeira regra, também, há uma restrição, definida no atributo “deny”, para não entrar em páginas de tópico para palavras derivadas da língua extraída em questão, isso é evidente pois os termos dentro de tal categoria serão palavras de outros idiomas que tem como origem alguma palavra da língua extraída em questão.

A segunda regra pegará links que levam até páginas de extração, em especial a última que restringe seguir link apenas dentro de tags que tenha como o atributo lang igual ao do idioma extraído. Essa regra, então, possui um callback para a função parse_word_page que é onde a extração dos dados ocorrem, também é colocado follow como true, isso porque, por padrão, é colocado em falso quando há um callback, assim após a extração dos dados, a aranha não seguiria os links encontrado na página de extração, por isso não se deve deixar em false. Por último, nesta segunda regra também foi adicionado um seletor XPath. Similarmente à seleção css, são selecionados elementos que tenham o atributo Lang do idioma que se deseja extrair, contudo, nesta seleção é assegurado que o elemento não deve possuir nodos ancestrais pertencentes a nenhuma das três classes: p-form-of, form-of ou NavFrame. Esta restrição existe para os dados ficarem mais limpos e com menos redundâncias.

As duas primeiras classes, p-form-of e form-of, são usadas em tags que possuem alguma forma de uma palavra (seja plural, conjugação de verbo, etc.), e a última é usado nas tabelas de conjugações de verbos. Assim evitando redundâncias de uma mesma palavra, em conjugações diferentes, aparecendo múltiplas vezes em diversas entradas do banco de dados de saída.

Como já comentado, o DSPD passa os itens extraídos por dois pipelines que trabalha sobre esses dados. O primeiro tem a função de limpar sujeiras que eventualmente surgem das páginas do Wiktionary, e o segundo para inserir os itens em um SQLite.

O primeiro pipeline, chamado simplesmente de DictBotPipeline, trata de remover resultados com eventuais erros existentes em páginas do Wiktionary, como o

“Lua error: not enough memory”, que é um erro decorrente da linguagem de script Lua, utilizada no Wiktionary, essa mensagem de erro apareceu em algumas entradas do dicionário nas primeiras fases de desenvolvimento do DSPD, então esse pipeline foi criado. Outrossim, alguns outros erros foram tratados, como erros de tradução, em que uma palavra não possuía tradução, e uma mensagem de erro aparecia nos resultados. Além de erros, algumas limpezas de dados são realizadas, tal como remover informações clonadas que ocorriam dentro do campo de definições.

O segundo pipeline trata da manipular um SQLite. Se o banco de dados não existir no momento da instanciação do pipeline, ele trata de criá-lo. Ao receber um item, já limpo pelo primeiro pipeline, será realizada a inserção no banco.

Para a execução do DSPD, faz-se necessário uma forma de salvar o progresso da extração, visto que a execução só acabaria quando todas as páginas de um idioma, alcançáveis pelas regras definida, fossem navegadas, o que pode ser muito longo. O próprio framework possui um sistema de salvar progresso, chamado de jobs. Basta definir um diretório para o job na hora de rodar o scraper que todo progresso será salvo neste diretório, e carregado dele em caso de já existir progresso salvo na hora da execução.

Também é necessário passar por linha de comando alguns argumentos de configuração, como o código ISO de dois dígitos do idioma de que se deseja extrair as definições, o caminho para o banco de dados que serão armazenados os dados extraídos, e por último, na versão mais recente do DSPD, vide Apêndice A, também é necessário passar o código do idioma para o qual será feita a tradução ou definição das palavras.

Um exemplo de linha de comando para rodar a extração de um dicionário de inglês para português ficaria como no trecho de código a seguir:

```
scrapy crawl dict_bot -O en_ptDict/en_ptDict1.json -a db_name="en_ptDict/en_ptDict"
-a lang=en -a base_lang=pt -s JOBDIR=crawls/dictBot_en_pt-1
```

O argumento “lang=en” indica que o idioma a ser extraído é o inglês, “base_lang=pt” indica que a extração será feita para o português, “db_name="en_ptDict/en_ptDict"” indica o caminho e nome ao banco de dados, e JOBDIR serve para apontar onde se deve salvar o progresso (job) realizado na extração. A opção “-O” é para gerar uma saída, neste caso em JSON.

Também é possível mexer nas configurações do DSPD no arquivo settings.py, contudo é recomendado que não se remova completamente o tempo de atraso das requisições, pois isso pode gerar irritação do servidor do Wiktionary, correndo riscos, em casos mais graves, de banimento de IP.

3.2. INTLIN

O INTLIN é o produto principal proposto por esse trabalho, ele surge com a intenção de integrar diferentes recursos usados em estudos de idiomas, como já discutido na seção de introdução. Seu uso é pensado para estudantes de nível intermediário em um idioma,

o qual já possuem um entendimento da língua estudada, e desejam ampliar seu vocabulário ou observar a língua sendo usada em contextos mais realistas, por meio de alguma mídia de seu interesse, tendo acesso a um sistema de flashcards e dicionários em tempo real durante a utilização da aplicação, sem necessitar parar de consumir a mídia estudada, de modo a otimizar o tempo de estudo.

No INTLIN é utilizada uma arquitetura MVC para que haja separação de responsabilidades. A parte da view contém tudo relacionada a GUI, i.e., as especificações em FXML dos fragmentos das interfaces gráficas e seus viewController, que são controladores da interface gráfica usado pelo JavaFx, eles são responsável por manter variáveis de estado, ações de botões, tratamento de eventos ou qualquer lógica necessária a interface gráfica. Cada fragmento da view pode ter no máximo um controlador. Para o controller têm-se controles mais gerais, que mantém as configurações gerais do sistema, variáveis de estados globais e também é responsável por se comunicar com o modelo para conseguir resultados requisitados pelo usuário na view. O modelo possui as lógicas para os módulos de dicionários e sistema de repetição espaçada.

O Modelo do sistema contém a lógica principal da aplicação, fazendo a comunicação e manipulação de bancos de dados, por meio das lógicas apropriadas. O modelo do sistema comporta dois grandes módulos, o módulo de dicionários e o módulo de repetição espaçada (ou módulo SRS).

O módulo de dicionário contém as lógicas do dicionário, é usado principalmente para consultas de palavras, mas também há funcionalidades de alterações de dados no banco de dados dos dicionários. Também é responsabilidade desse módulo o parsing de arquivos para inserção em banco de dados SQLite. Este módulo é projetado de modo a ser expansível para que outros esquemas de banco de dados de dicionários possam ser utilizados no futuro.

Todo tipo de dicionário no sistema deve estender uma classe abstrata chamada de Dictonary, ela define uma API comum para os dicionários, bem como provê métodos genéricos para conexão com os bancos SQLite. A ideia é deixar o sistema expansível para que outros tipos de dicionários, com esquemas de dados diferentes, possam ser implementados no futuro, bastando que esses novos dicionários sigam a API mínima definida. Todo dicionário necessita também de um parser que converte seu arquivo original para um Banco de dados que pode ser usado pelo sistema, para o caso do novo dicionário só possuir algum arquivo, como JSON ou XML.

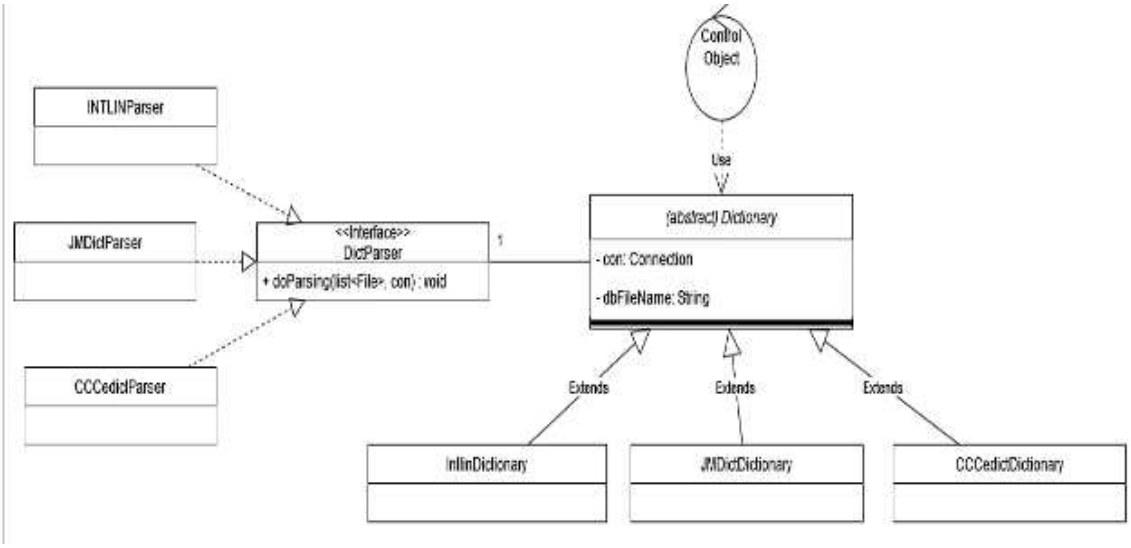


Figura 12. Diagrama de classes - dicionários

A Figura 12 contem uma visão simplificada de um diagrama de classes do modelo, tendo os métodos das classes de dicionários ocultados. A classe abstrata Dictionary contém métodos genéricos para realizar conexão com um SQLite, e uma gama de métodos abstratos a serem estendidos pelas classes filhas, que são métodos mínimos que qualquer outro tipo de dicionário que for integrado ao sistema deve contemplar, como procurar definições e realizar alterações na base de dados.

A classe de Dictionary é composta por um parser, o DictParser, que é uma interface que expõe apenas uma funcionalidade, a de realizar o parsing sobre um arquivo. A ideia desse parsing é para transformar algum tipo de arquivo de dicionário em um banco de dados que pode ser usado por uma classe de dicionário, para caso esse banco de dados não exista. Por exemplo, têm-se vários arquivos JSON gerado pelo DSPD, mas por algum motivo o banco de dados se perdeu ou não foi gerado na hora da extração, o IntlinParser converterá esses arquivos JSON para SQLite.

Também é possível perceber na Figura 12 que, além do dicionário padrão do INTLIN (que usa o mesmo modelo de dados do DSPD), existem dois outros tipos de dicionário, o CC-cedict (Mandarim) e o JMdict (Japônes), estes dicionários não estão completamente integrados ao sistema, não sendo possível utilizá-los no protótipo.

O módulo de SRS é responsável pela criação e administração dos flashcards, tendo a responsabilidade de definir datas de revisões de cartões estudados de acordo com o feedback do usuário e do nível de proficiência com dado cartão.

O sistema para repetição espaçada usado no INTLIN é baseado no Sistema de Leitner, que é um algoritmo que utiliza cinco grupos de proficiência no qual os cartões podem se enquadrar. Quanto mais proficiência um usuário tiver em um cartão, mais espaçadas as revisões desse cartão serão.

No INTLIN, quando o usuário revisa seus cartões, ele pode responder que o conteúdo de um cartão foi: (i) Impossível de lembrar/falhou. (ii) Difícil de lembrar. (iii) Conseguiu lembrar. (iv) Fácil de lembrar.

De acordo com seu nível de proficiência no cartão e com sua resposta, uma data de próxima revisão será calculada. Para caso de falha, isto é, a não lembrança do conteúdo do cartão, a próxima revisão é agendada para o dia atual e o nível de proficiência é reiniciado para o mais baixo. Respostas difíceis não causam mudança de

proficiência mas geram uma data de revisão próxima do dia atual. Respostas boas aumentam a proficiência por um nível e agendam a revisão para datas um pouco mais espaçadas. Por último, respostas fáceis aumentam proficiência por dois níveis e geram repetições bem mais espaçadas.

Também existe um fator de ease, que é aumentado com respostas fáceis ou, dependendo da proficiência, boas. Esse fator pode dar uns dias extras no agendamento da revisão, caso esteja em um valor alto o suficiente. Ele existe para que em caso de errar um cartão, não gere uma resposta tão punitiva para quem no passado já o tinha aprendido.

No calculo para a agendar uma data de revisão é utilizado o padrão de projeto estratégia, no qual a estratégia para o cálculo é escolhida baseada na dificuldade informada pelo usuário para o cartão.

Similar aos dicionários, os cartões do usuário são mantidos localmente em um SQLite. Seu esquema é apresentado pela Figura 13.

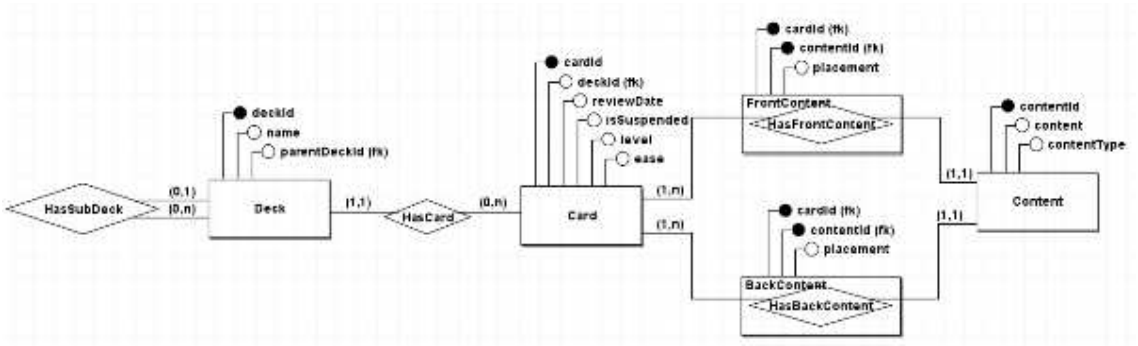


Figura 13. Esquema banco de dados SRS

Em suma, os cartões possuem as informações principais para a lógica de calcular prazos de revisões: level (proficiência), ease e a data de revisão atual em si. Os cartões podem ter múltiplos conteúdos na frente e no seu verso, e cada conteúdo pertence unicamente a um cartão, isso pode ser observado pelas cardinalidades na tabela associativa, um cartão pode estar várias vezes dentro de tais tabelas, enquanto cada conteúdo aparecerá uma única vez, i.e. junto ao seu cartão. Cada cartão pertence unicamente a um baralho, que por sua vez pode ser um “sub-deck” de outro baralho, e.g. o baralho de “conjugações de verbos” é um sub-deck do baralho “gramática”. Os atributos placement das tabelas associativas e contentType de content servem para controles dentro da aplicação.

Os cartões a serem revisados no dia são carregados em memória em uma classe do modelo, onde podem ser calculadas todos os pontos relacionados ao próximo agendamento de revisão, seguindo as regras comentadas anteriormente.

A parte de controladores do sistema dispõe de dois controladores globais, um controlador principal que administra informações gerais de estado do sistema, e um controlador de configurações que é responsável por trabalhar com o arquivo de configuração.

O Controlador global é uma classe singleton, que é majoritariamente chamada pela camada de visão (View) para que alguma requisição ao modelo seja feita. Bastante

informação de estado do sistema também são salvas nesta classe: idiomas suportados, idioma selecionado, dicionários carregados, listas de cards para revisão do dia, entre outros.

Por muitas dessas variáveis serem do estado global do sistema, alterá-las pode implicar que seja necessário executar códigos em outras camadas da aplicação. Por isto, é utilizado no controle o padrão observer, no qual o controlador global implementa uma interface chamada “Observable” e poderá notificar classes observadoras que se inscrevam no controlador. De modo geral, os observadores serão viewControllers, e as notificações serão usadas para atualizar a interface gráfica de algum modo.

O sistema utiliza um arquivo de configurações em XML, que, no momento, mantém duas configurações: uma data (usada para uma lógica interna) e o último idioma carregado para estudar. Este controlador trabalha com leitura e escrita do arquivo de configurações, utilizando a classe utilitária Properties, nativa do Java.

O papel deste controlador é pequeno, sendo utilizado para recuperar e escrever informações de configurações no arquivo, também é usado na inicialização da aplicação, quando é lido o arquivo de configuração e salvas todas as informações em memória.

As interfaces gráficas do sistema foram implementadas com a plataforma JavaFx, junto a ferramenta SceneBuilder que permite a fácil construção de layouts via drag and drop, evitando trabalhar diretamente com o arquivo estilo XML que o JavaFx utiliza.

Primeiramente, desenvolveu-se a tela inicial, tendo como principal consideração que o sistema de dicionários deve estar acessível durante o estudo da mídia. Desta forma, os dicionários ficam dispostos no inferior da tela em uma área expansível, deste jeito, é possível esconder os dicionários durante o estudo de algum conteúdo, e trazê-los à vista quando houver necessidade. Acima desta região expansível dos dicionários, está disposto um painel de tabs, em que é possível alternar entre o módulo de mídia e do sistema de SRS, o motivo de tal separação se dá pelo fato que, de modo geral, os dois não serão usados ao mesmo tempo, sendo a aba de mídia usada para estudos e a de SRS para revisões. Assim que o sistema é aberto, tem-se na aba de mídias um botão em formato de ícone para carregar um arquivo, além disso, não há muita informação para apresentar na tela, então visando aproveitar um pouco do espaço disponível, foram inseridos regiões com informações rápidas sobre o uso do sistema.

A aba de SRS, também não contem nenhuma informação para apresentar quando não há um idioma carregado para estudo no sistema, então para ocupar a área inutilizada, também foi inserido um helper text.

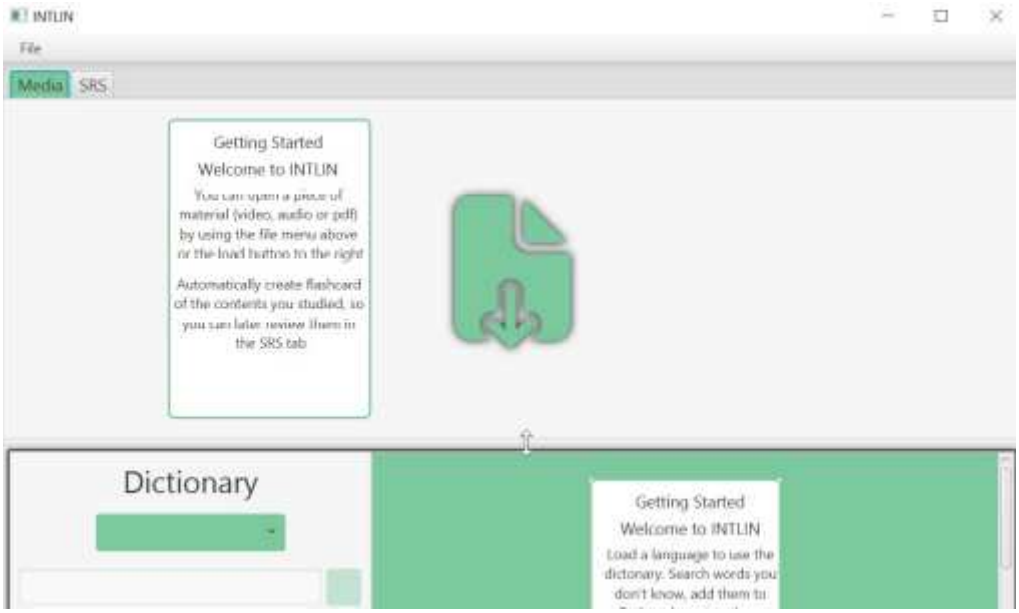


Figura 14. Tela principal INTLIN

Figura 15. Aba de SRS vazia

Para a região expansível de dicionários, foi feita uma divisão em duas colunas, a primeira é responsável por selecionar o dicionário e realizar consultas, enquanto a segunda tem a responsabilidade de apresentar o resultado da busca. É possível escolher um idioma pelo menu file ou pela caixa de seleção na região de dicionários, após selecionar o idioma desejado, é possível realizar consultas no dicionário.

A Figura 16 apresenta uma consulta realizada para a palavra “Álcool” em um dicionário de português para Inglês. Também é possível observar, um botão para salvar

uma entrada como um flashcard em um baralho de revisão, bem como regiões para os outros dados que foram extraídos do Wiktionary, tais quais sinônimos e antônimos.

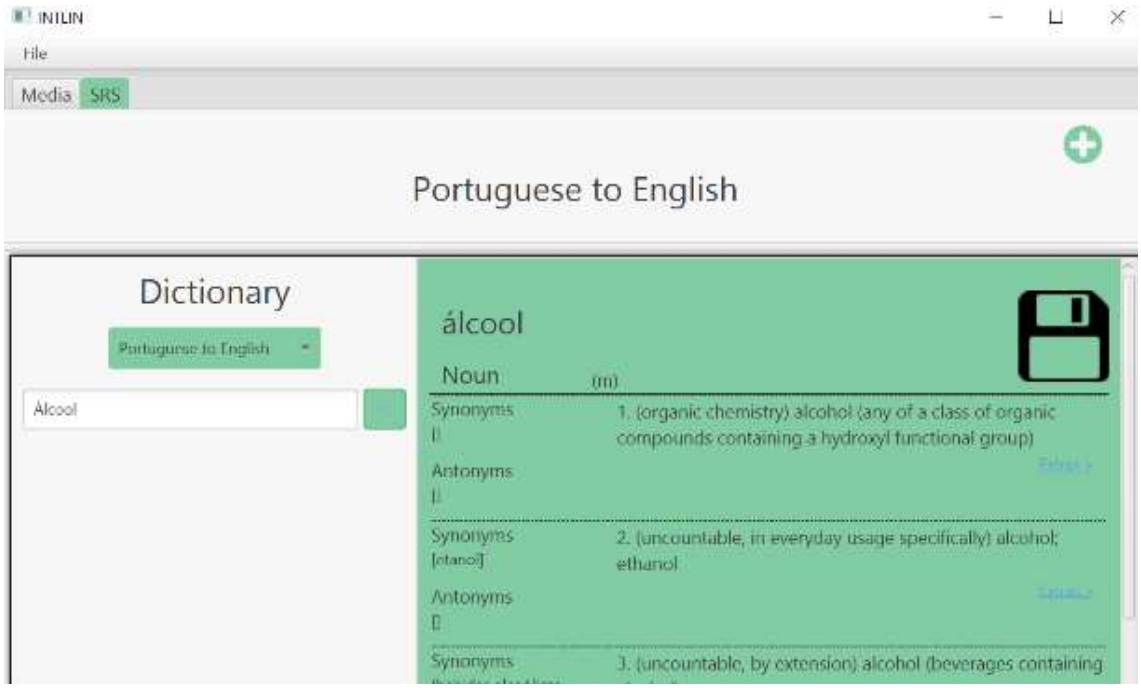


Figura 16. Pesquisa no dicionário

Na aba de mídias, é apresentado um ícone clicável em formato de arquivo, como já visto. A partir dele, é possível abrir arquivos de áudio, vídeo ou PDF. Carregar áudio ou vídeo levará a uma mesma tela, sendo a diferença que para áudio, onde seria exibido o vídeo, é apresentado um ícone centralizado de headphones. PDF abre uma webview junto com um plugin de visualização de PDF.

Ao carregar áudio ou vídeo, duas seções são renderizadas na tela, a principal com o vídeo e barra de progresso e uma outra onde ficará a transcrição da legenda, como podemos observar na Figura 17. Na região da transcrição de legenda há um botão no topo para se carregar uma legenda, que deve ser um arquivo de extensão srt. Na parte inferior à transcrição é possível visualizar um botão desativado, ele permanecerá desativado enquanto não houver uma legenda ou um idioma carregado no sistema, e ele é utilizado para salvar trechos selecionados da transcrição de legendas em um flashcard que poderá ser eventualmente revisado pelo usuário. A Figura 18 explicita o sistema com legendas carregadas, bem como algumas selecionadas.

Em ambas as Figuras também está presente um botão de câmera sobre o vídeo, este botão realiza uma captura de tela do vídeo e adiciona em um flashcard. Na primeira Figura, ele se encontra levemente transparente, isso ocorre para não atrapalhar a visualização da mídia em execução. Ao passar o cursor por cima, um efeito de highlight ocorre evidenciando o botão, como possível ver na Figura 18. Quando um dos dois botões são acionados, uma janela aparecerá pedindo para que o usuário adicione uma solução ao flashcard e confirme sua criação, vide a Figura 19.

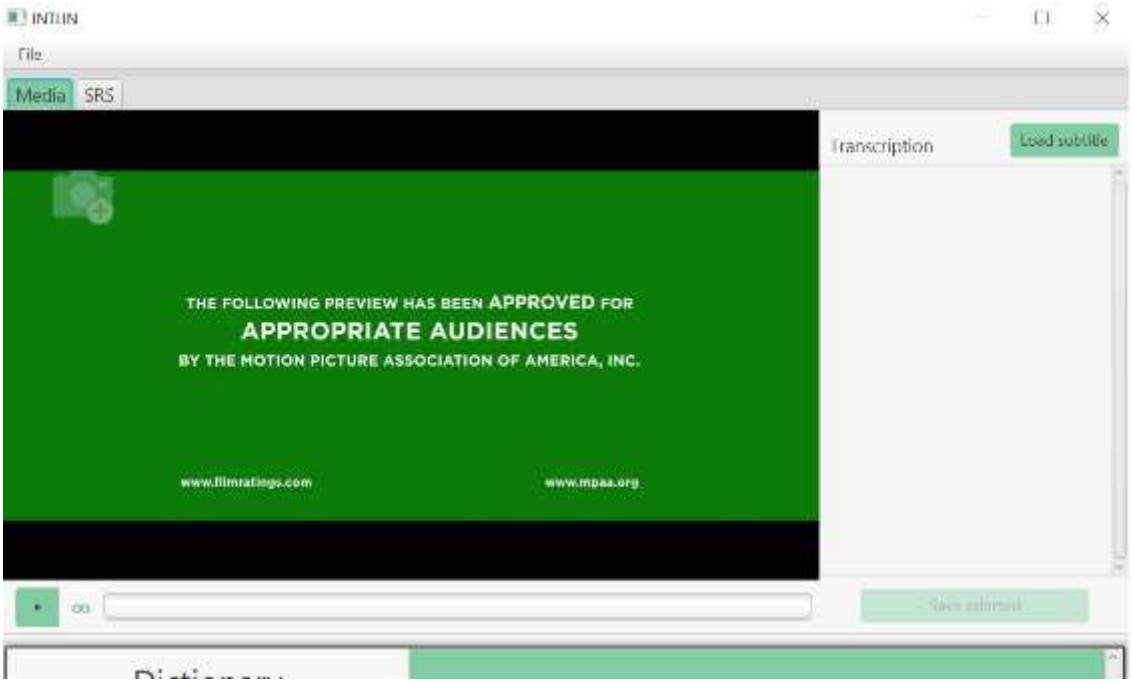


Figura 17. Vídeo aberto no INTLIN

Figura 18. Botões na aba de mídia

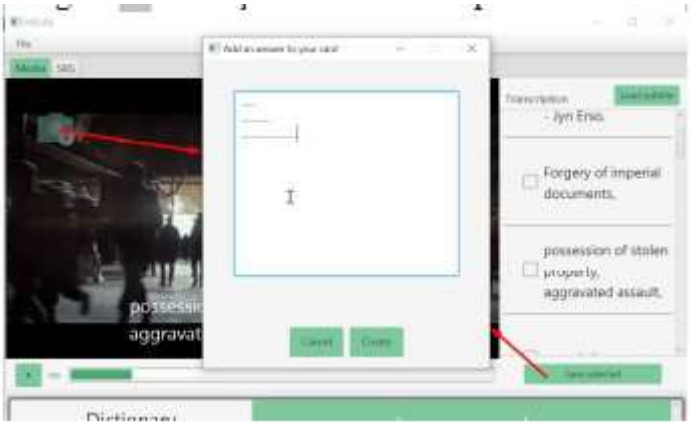


Figura 19. criação de flashcard a partir de mídia

Para a exibição de PDFs foi usado uma webview do JavaFx junto com um plugin PDF.js1. A Visualização de PDF, apesar de funcional, possui alguns bugs, um visual em que os botões do leitor não aparecem corretamente renderizados, e um segundo bug mais grave de causa não identificada em que a aplicação inteira fica lenta e pode levar até a crashes no sistema. O bug visual pode ser averiguado na Figura 20, em que mostra um PDF aberto no INTLIN. Também é possível carregar um arquivo de áudio através do botão “Load audio”, em que o sistema permite ao usuário ouvir áudio enquanto lê o texto, para funcionalidade de audiobook. Áudio e PDF podem ser vistos simultaneamente carregados na Figura 21.

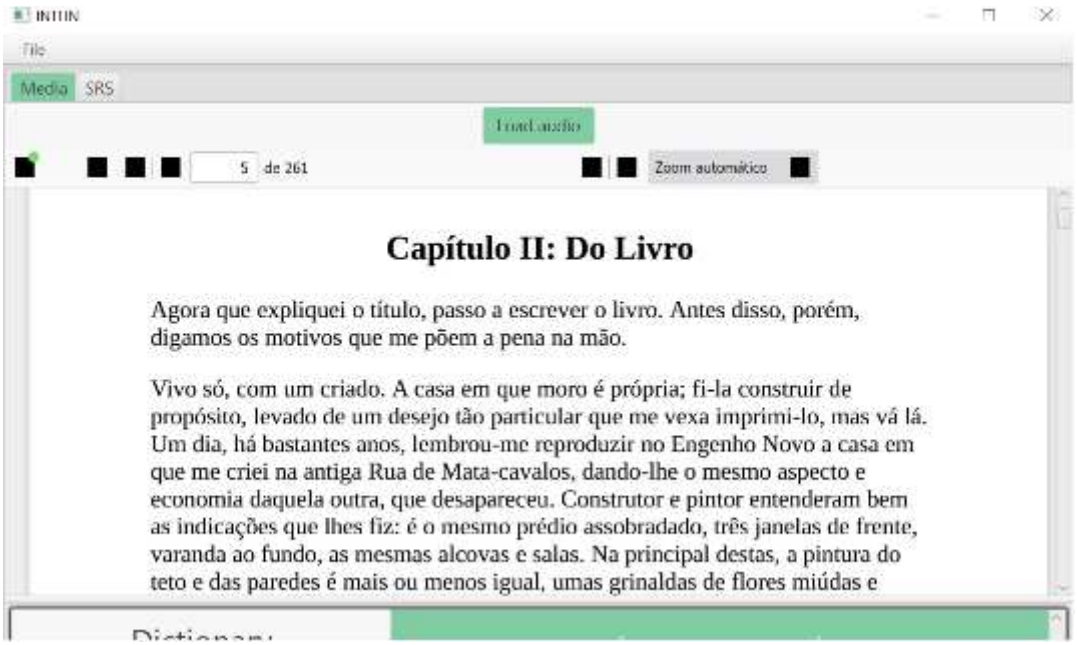


Figura 20. PDF aberto no INTLIN

Figura 21. PDF e áudio abertos no INTLIN

Por último, a aba de SRS apresenta o nome do idioma selecionado para estudos como cabeçalho, um botão para criação de novos baralhos localizado no topo, o nome de todos os baralhos existentes, para cada baralho há um botão para criação de cartões, bem como uma contagem de revisões agendadas para o dia. Vide Figura 22.

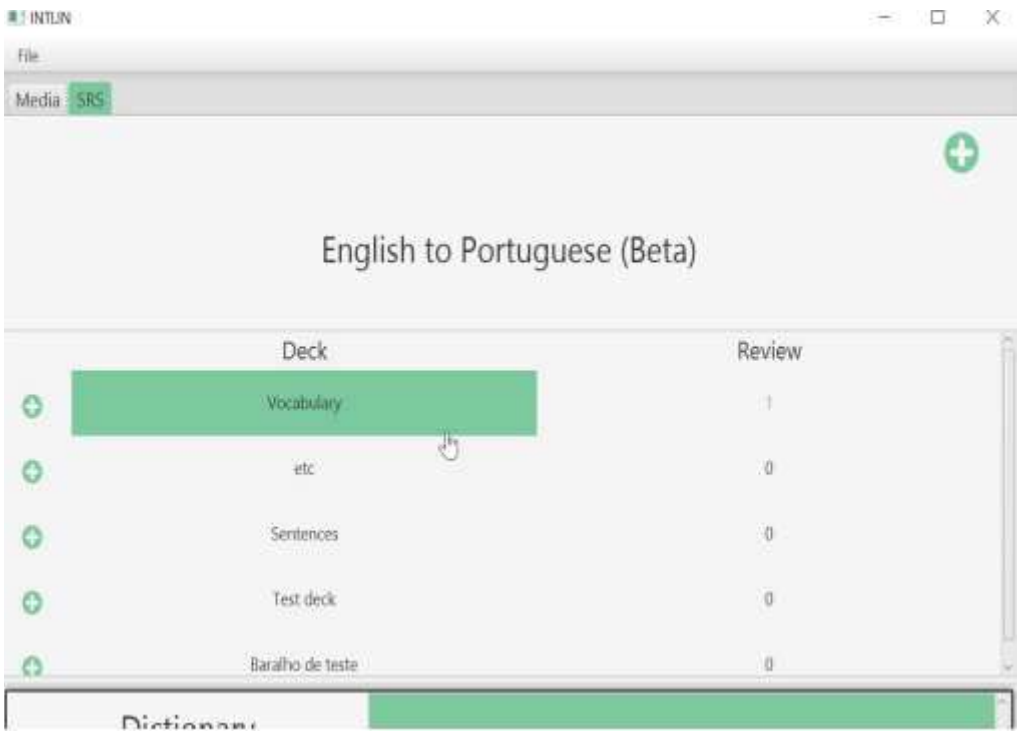


Figura 22. Aba de SRS

Ao clicar no botão de adição ao lado de um baralho, uma janela para criação de um cartão é aberta, ao final deste processo, o cartão será adicionado ao baralho

designado, a Figura 23 demonstra a criação de um cartão manualmente.

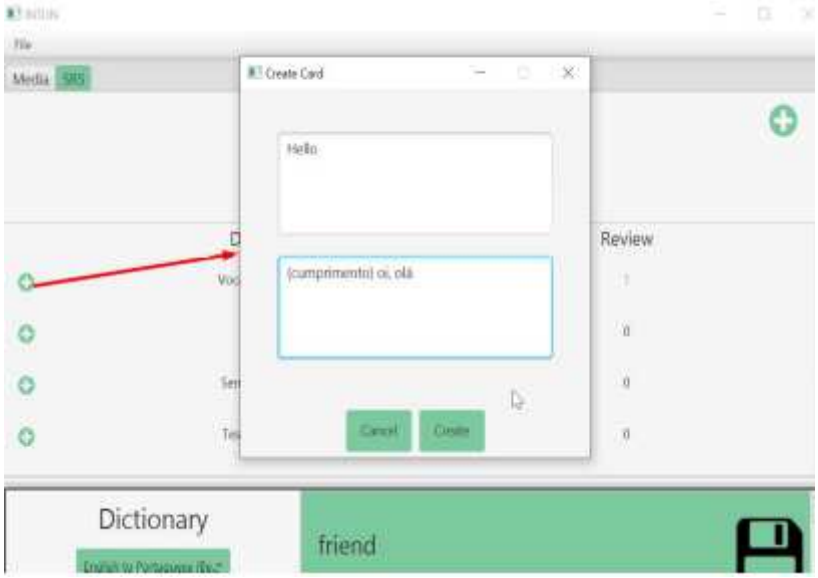


Figura 23. Criação de cartão manualmente

Quando o usuário clicar em um baralho, abrirá uma janela de revisões, a qual será exibido, sequencialmente, todos os cartões que o usuário tem agendado para o dia. Tais cartões são capazes de exibir textos e imagens, e podem ser criados manualmente como visto acima, via log de legendas e por screenshot de um vídeo carregado no sistema. A Figura 24 é referente a um cartão aberto para revisão, contendo ambos texto e imagem, na parte superior da tela tem o conteúdo a ser revisado, abaixo disso tem uma região escondida que aparecerá a resposta, e na parte inferior da tela estão dispostos os quatro botões para o usuário auto avaliar o quão bem ele se lembra do conteúdo. Quando a revisão de um cartão é finalizada, a resposta do mesmo é exibida e é apresentado o feedback da data que o cartão estará disponível novamente para estudo, vide Figura 25.

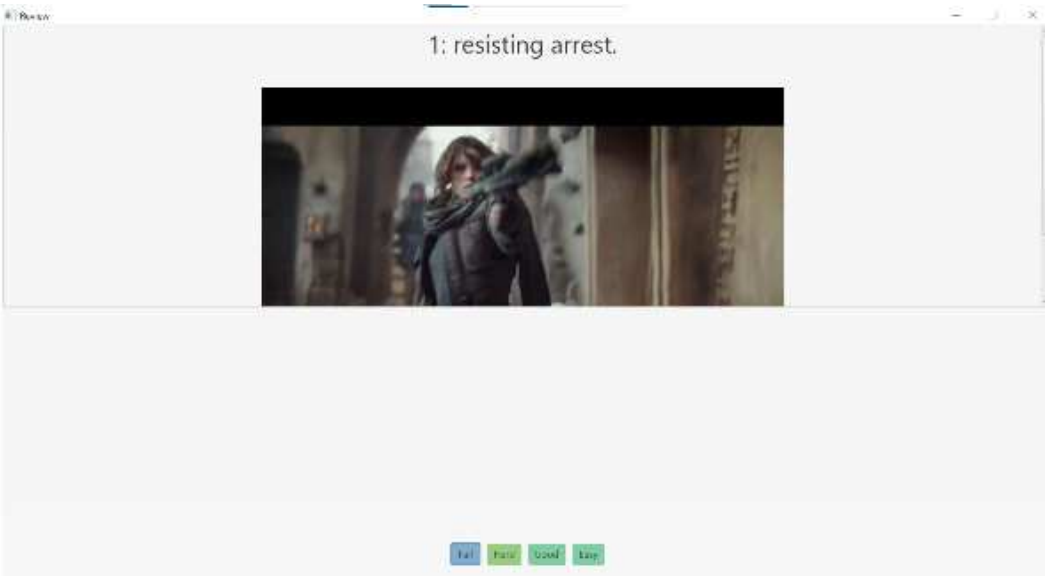


Figura 24. Revisão de flashcard

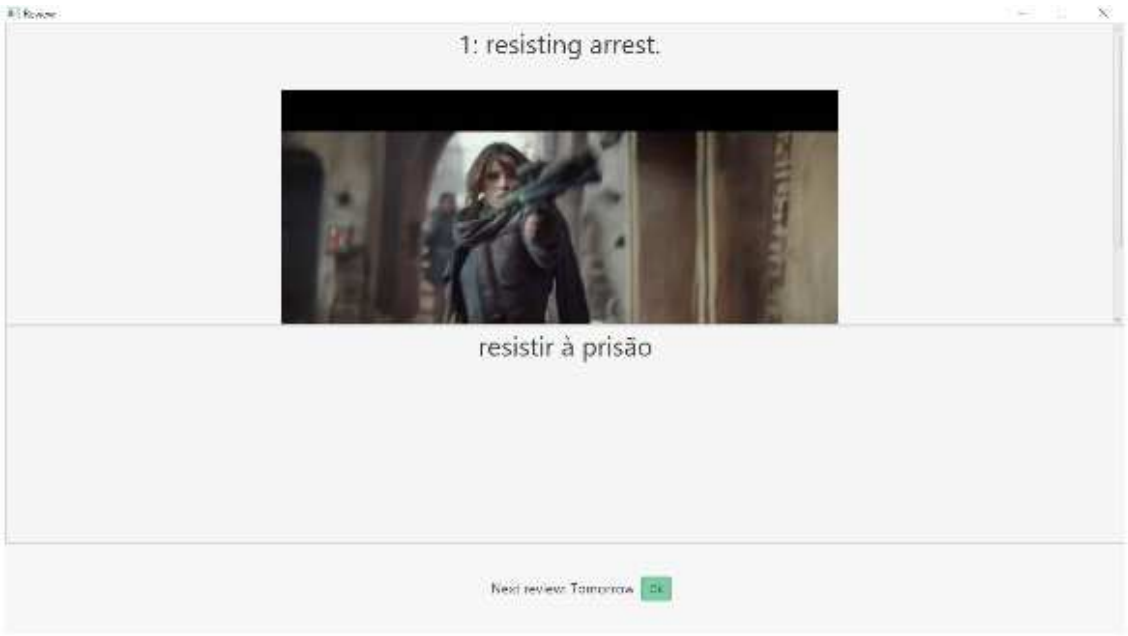


Figura 25. Revisão de flashcard finalizada

O protótipo definiu as principais funções do sistema, que tiveram seu desenvolvimento relatado na seção anterior. Contudo, ainda existem algumas funcionalidades que ficaram de fora desta primeira versão, dentro do modelo existem algumas funcionalidades implementadas que não foram portadas na interface gráfica do sistema, sendo impossível o usuário realizar tais ações, e.g., alterar valores nos bancos de dados dos dicionários, suspender um cartão, criação de subdecks, dentre outros.

Dentro do protótipo, existe um bug com causa não identificada quando há um PDF carregado, como já mencionado, o problema é relativamente recorrente, o que pode tornar o estudo com PDF um pouco frustrante ao usuário. Não se consegui traçar uma causa para tal bug, assim o mesmo não foi corrigido. Fora isso, as demais

funcionalidades constadas no protótipo funcionam como esperado.

4. Avaliação do protótipo

Após concluir o desenvolvimento do protótipo, surge a necessidade de uma avaliação com alguns usuários. Então, para a avaliar o protótipo, definiu-se um teste de usabilidade para se realizar em ambiente controlado com três participantes anônimos. Os três são bons membros do público-alvo, pois um gosta de estudar idiomas como hobby e os outros dois são estudantes de Inglês.

Para o teste, planejou-se um conjunto de atividades para os participantes realizarem, e para cada uma delas levantou-se questões de análise quanto à eficácia e eficiência do sistema, isto é, se a porcentagem esperada dos participantes conseguiram concluir a tarefa e se as tarefas foram concluídas dentro do tempo esperado. Após realizarem as atividades, a fim de medir o nível de satisfação dos participantes do teste, o questionário System Usability Scale (SUS) foi aplicado e ficou aberto um espaço para comentários sobre o protótipo.

O objetivo do teste é para que se obtenha feedback de usuários, para poder analisar se as escolhas realizadas na interface estão boas em termos de usabilidade, podendo traçar um plano de onde prosseguir e do que melhorar.

Levantou-se 10 atividades para o teste e para cada uma delas foi definido a porcentagem de participantes que se espera que consigam concluir a atividade, bem como a quantidade máxima de tempo esperada para a conclusão. As atividades selecionadas, bem como as métricas esperadas estão dispostas na Tabela abaixo. Nesta tabela, estão dispostas as atividades selecionadas para realização no teste, bem como os requisitos de usabilidade para eficácia e eficiência, por exemplo, para a Atividade 1 espera-se que cada um dos três participantes consiga concluí-la em um tempo de até 18 segundos.

Para medir níveis de satisfação, também foi realizado o questionário SUS com os participantes, cujo o objetivo é ter pontuação acima de 75 pontos, que é o mínimo considerado bom. Abaixo disto, é um forte indicativo que o design do sistema precisa ser repensado ou refatorado em alguns aspectos.

#	Atividade	Eficácia	Eficiência
1	Selecionar um idioma para estudo no sistema	3/3	18 segundos
2	Carregar uma mídia de vídeo para estudar	3/3	24 segundos
3	Carregar uma legenda para o vídeo carregado	3/3	24 segundos
4	Usar o dicionário para pesquisar alguma palavra	3/3	18 segundos
5	Salvar uma entrada do dicionário em um flashcard	3/3	9 segundo
6	Salvar uma captura de tela em um flashcard	3/3	27 segundos

7	Salvar trechos da transcrição de legenda em um flashcard	3/3	27 segundos
8	Abrir um baralho e revisar um cartão	3/3	30 segundos
9	Fechar a mídia de vídeo aberta	2/3	21 segundos
10	Carregar PDF + áudio (audiobook)	3/3	58 segundos

Tabela 2. Atividades do teste

Para termos de eficiência, as atividades de carregamento de arquivos no sistema (2, 3 e 10) ganharam valores consideravelmente altos, pois são atividades em que se deve esperar o seletor de arquivos do sistema operacional abrir, pensar onde está o arquivo e navegar até o mesmo, a atividade 10 é o tempo de abrir dois arquivos somados com um tempo adicional para caso o usuário queira olhar o PDF antes de abrir um áudio (24 s + 24 s + 10 s). A atividade de revisão dos flashcards criados durante o teste também recebe um tempo esperado de execução elevado, levando em consideração diversos fatores, tais como o usuário terá de ler o cartão (algum pode ter bastante texto), o usuário pode marcar algum cartão como errado (Botão fail) múltiplas vezes, o que faria o cartão aparecer diversas vezes para revisão. Tarefas que envolvem digitar respostas de flashcards (e.g. salvar captura e trechos de legenda) receberam um tempo generoso também, caso os participantes queiram digitar alguma informação.

Quanto a eficácia, acreditava-se que a única atividade que poderia gerar algum problema seria a 9ª, quanto ao fechamento de uma mídia carregada, pois a ação para fechar uma mídia carregada não está à vista nas telas principais onde o usuário já está focado e sim está escondido como um botão de menu dentro do menu file, essa desconfiança por si só já pode indicar um problema com este design. Desta forma, colocou-se que um dos três participantes do teste podem vir a falhar nesta atividade.

Com cada um dos participantes, a execução do teste iniciou com a afirmação de que caso não consigam realizar algumas das atividades requisitadas a culpa não é dos participantes e sim do design utilizado no protótipo, e que esse teste é justamente para avaliar a usabilidade e qualidade do design adotado no sistema. O teste foi realizado presencialmente, e os participantes aceitaram em realizar o teste de forma anônima.

Após um breve explicação sobre o sistema, seguiu-se para as atividades. As Tabelas abaixo trazem os resultados por atividade e o resultado do questionário SUS. Os valores foram preenchidos em vermelho quando não se alcançou o valor estipulado e em verde caso contrário.

#	Atividade	Eficácia esperada	Eficácia alcançada	Eficiência	Eficiência alcançada
1	Selecionar um idioma para estudo no sistema	3/3	3/3	18 segundos	3/3
2	Carregar uma mídia de vídeo para estudar	3/3	3/3	24 segundos	3/3

3	Carregar uma legenda para o vídeo carregado	3/3	3/3	24 segundos	3/3
4	Usar o dicionário para pesquisar alguma palavra	3/3	3/3	18 segundos	3/3
5	Salvar uma entrada do dicionário em um flashcard	3/3	3/3	9 segundo	1/3
6	Salvar uma captura de tela em um flashcard	3/3	3/3	27 segundos	3/3
7	Salvar trechos da transcrição de legenda em um flashcard	3/3	3/3	27 segundos	3/3
8	Abrir um baralho e revisar um cartão	3/3	2/3	30 segundos	2/3 (uma desistência)
9	Fechar a mídia de vídeo aberta	2/3	3/3	21 segundos	1/3
10	Carregar PDF + áudio (audiobook)	3/3	3/3	58 segundos	3/3

Tabela 3. Síntese do teste

Pontuação SUS	Pontuação
Participante 1	100 pontos
Participante 2	72.5 pontos
Participante 3	75 pontos

Tabela 4. Resultados SUS

A atividade 8 foi a única que realmente falhou em termos de eficácia, e foi devido ao fato do participante não entender que SRS, Spaced Repetition System, tratava-se de revisões, talvez a utilização de um termo menos técnico na aba de SRS, como “Review”, poderia ter ajudado a evitar esse desentendimento. Ao contrário do esperado, ninguém falhou na atividade 9, sobre fechar uma mídia aberta.

Para eficiência, pode-se observar na atividade 5, Salvar uma entrada do dicionário em um flashcard, que os participantes tiveram dificuldades para entender como salvar uma entrada do dicionário em um flashcard, o feedback comum recebido foi que o botão parecia uma decoração, e acabou sendo ignorado. Para a atividade 9,

apesar de todos terem conseguido realizá-la, apenas um deles terminou dentro do tempo estimado, o problema principal se dava ao fato da ação de fechar mídia estar um pouco escondido.

Finalmente, para satisfação, um dos participantes rendeu uma pontuação abaixo dos esperados 75 pontos, e os outros dois renderam pontuação maior ou igual. Contudo, o participante 1 rendeu uma pontuação de 100 pontos, mas foi possível observar durante todo o processo de teste que o sistema não está em “um nível de 100 pontos”, então é bom considerar as respostas do SUS do participante 1 com um pouco de cautela.

De modo geral, o teste gerou um ótimo feedback para saber em que pontos o sistema pode melhorar, mas sempre importante ter em mente que a amostra não era muito grande, e que então a confiabilidade dos resultados não é de 100% (vide questionário SUS com mais de 100 pontos).

5. Conclusão e considerações finais

Neste trabalho, desenvolveu-se um protótipo de um sistema para auxílio na aprendizagem de idiomas estrangeiros, que utiliza a abordagem comum de imersão em um idioma por meio de mídias nativas e autênticas, paralelamente a fazer anotações para salvar em flashcards para revisões automatizadas e otimamente espaçadas a fim de tirar proveito do spacing effect (Ebbinghaus, 1885). O protótipo desenvolvido visa otimizar o tempo de estudo possibilitando criação rápida de flashcards por meio de poucos cliques de botões bem como disponibilizando um sistema de dicionário na própria plataforma, de modo a agilizar o processo de pesquisa por palavras desconhecidas. Assim, foi desenvolvido um protótipo com três módulos principais: para mídia, para dicionário e de repetição espaçada (SRS), sendo possibilitado pelo sistema um bom nível de comunicação entre eles, visando otimizar o tempo de estudo.

Desenvolveu-se, também, um extrator de dicionário que busca definições no Wiktionary, o extrator é consideravelmente genérico, podendo extrair definições de múltiplos idiomas. Os dados extraídos durante o trabalho foram utilizados no módulo de dicionário do INTLIN, contudo o DSPD é um sistema a parte e pode ser utilizado para outros fins.

5.1. Lições aprendidas

No início do projeto, definiu-se um escopo demasiadamente grande para o protótipo, com muitas funcionalidades, não havendo tempo de incluí-las no protótipo inicial, assim bastante do escopo foi cortado com o desenrolar do projeto. Também, houve funcionalidades planejadas que não foram possível por limitações das tecnologias escolhidas, e.g. assim como é possível extrair imagens de vídeos para adição em flashcards, desejava-se extrair trechos de áudios para salvar, contudo a API padrão de mídias do JavaFX não permitia tal ação, existiam algumas bibliotecas para tal, contudo bastante antigas e desatualizadas, gerando bugs inesperados, cortando-se assim essa funcionalidade do escopo.

Extraiu-se opiniões valiosíssimas de membros do público-alvo durante a realização do teste. Pode-se coletar diversos feedback de onde o sistema pode melhorar, tais como uma melhor indicação das revisões, uma forma mais eficiente em termos de usabilidade para salvar entradas de dicionários em flashcards, questões quanto ao log de legenda, menu file, dentre outros.

5.2. Possibilidades de trabalhos futuros

A partir deste trabalho, existem dois caminhos que se pode seguir de trabalho, um com o DSPD e outro com o INTLIN.

Para o INTLIN, caberia implementar mais funcionalidades e prestigiar melhorias necessárias vinda do teste, de modo a transformar o protótipo em uma versão com cara de “Produto finalizado”.

Para o DSPD, é possível utilizá-lo para qualquer motivo que se deseje informações linguísticas, e.g. para processamento de linguagem natural (PLN). É possível expandir o bot para outros idiomas, adicionar ou remover informações extraídas das páginas, ou qualquer alteração necessária.

6. References

KRASHEN, Stephen, The Input Hypothesis. issues and implications. Harlow: Longman, 1985.

Lightbown, Patsy M.; Nina Spada – How Languages Are Learned (Oxford Handbooks for Language Teachers). 3ª edição. Oxford: Oxford University Press, 2006.

Lyu, Dayan, A Critical Review of Krashen’s Input Hypothesis: Three Major Arguments. Estados Unidos da América: American Research Institute for Policy Development em Journal of Education and Human Development, 2015.

Brown, H. D. – Principles of Language Learning and Teaching. 5ª edição. Londres: Pearson Education ESL, 2006.

McLaughlin, B, Theories of Second Language Learning. London: Edward Arnold, 1987.

Garcia, Damaris, Spaced Learning: Its Implications in the Language Classroom. Costa Rica: Revista de Lenguas Modernas, 2014.

O que é o Duolingo?, Duolingo, 2020. Disponível em: <https://support.duolingo.com/hc/pt-br/articles/204829090-O-que-%C3%A9-o-Duolingo->. Acesso em: 04/05/2022.

Grygo, Agnieszka; Gajek, Elżbieta, Risks of Using Duolingo by Polish Learners at Primary Level. Países Baixos: EnetCollect WG3 & WG5 Meeting, 2018.

Aoki, Yuta. Japanese Guy Tries Duolingo Japanese. Youtube, 20 Abr. 2022, 2022.

Yabla, Yabla, 2022. Disponível em: <https://www.yabla.com/>. Acesso em: 09/05/2022.

Gamma, Erich; Helm, Richard; Johnson, Ralph; Vlissides, John – Design Patterns: Elements of Reusable Object-Oriented Software. 1ª edição. Boston: Addison-Wesley, 1994.

Kent, Beck – Test-Driven Development By Example. Boston: Addison Wesley, 2002.

Wazlawick, Raul Sidnei – Engenharia de Software – Conceitos e Práticas. 2ª edição. Florianópolis: GEN LTC, 2019.

Sirisuriya, SCM de S, A Comparative Study on Web Scraping. Ratmalana: Proceedings of 8th International Research Conference, KDU,, 2015.

Items, Scrapy Documentation, 2008-2022. Disponível em: <https://docs.scrapy.org/en/latest/topics/items.html>. Acesso em: 25/06/2022.

Selectors, Scrapy Documentation, . Disponível em: <https://docs.scrapy.org/en/latest/topics/selectors.html>.. Acesso em: 25/06/2022.

Item Pipeline, Scrapy Documentation, 2008-2022. Disponível em: <https://docs.scrapy.org/en/latest/topics/item-pipeline.html>. Acesso em: 25/06/2022.

Brooke, John, SUS: A quick and dirty usability scale. Earley: , 1995.

AutoThrottle extension, Scrapy Documentation, 2008-2022. Disponível em: <https://docs.scrapy.org/en/latest/topics/autothrottle.html>. Acesso em: 08/03/2022.

Spiders, Scrapy Documentation, 2008-2022. Disponível em: <https://docs.scrapy.org/en/latest/topics/spiders.html>. Acesso em: 08/03/2022.

Ebbinghaus, Hermann – Memory: A Contribution to Experimental Psychology. New York: Teachers College, 1885.