**­­FACULDADE DE TECNOLOGIA DE SÃO JOSÉ DOS CAMPOS**

**FATEC PROFESSOR Jessen Vidal**

**CESAR AUGUSTO SIQUEIRA SANTOS**

**QuizFATEC: Banco de Questões e Aplicativo Móvel Simulador dos Vestibulares da FATEC**

São José dos Campos

2020

**CESAR AUGUSTO SIQUEIRA SANTOS**

**QuizFATEC: Banco de Questões e Aplicativo Móvel Simulador dos Vestibulares da FATEC**

Trabalho de Graduação apresentado à Faculdade de Tecnologia de São José dos Campos, como parte dos requisitos necessários para a obtenção do título de Tecnólogo em Banco de Dados.

**Orientador: Me. Diogo Branquinho**

São José dos Campos

2020

**Dados Internacionais de Catalogação-na-Publicação (CIP)**

**Divisão de Informação e Documentação**

SANTOS, Cesar Augusto Siqueira

QuizFATEC: Banco de Questões e Aplicativo Móvel Simulador dos Vestibulares da FATEC.

São José dos Campos, 2020.

60f. (número total de folhas do TG)

Trabalho de Graduação – Curso de Tecnologia em Banco de Dados.

FATEC de São José dos Campos: Professor Jessen Vidal, 2020.

Orientador: Me. Diogo Branquinho.

1. Banco de Questões. 2. Raspagem de dados. 3. Graduação. I. Faculdade de Tecnologia. FATEC de São José dos Campos: Professor Jessen Vidal. Divisão de Informação e Documentação. II. Título

**REFERÊNCIA BIBLIOGRÁFICA**

SANTOS, Cesar Augusto Siqueira **QuizFATEC: Banco de Questões e Aplicativo Móvel Simulador dos Vestibulares da FATEC.** 2020. 60f. Trabalho de Graduação - FATEC de São José dos Campos: Professor Jessen Vidal.

**CESSÃO DE DIREITOS**

NOME(S) DO(S) AUTOR(ES): Cesar Augusto Siqueira Santos

TÍTULO DO TRABALHO: QuizFATEC: Banco de Questões e Aplicativo Móvel Simulador dos Vestibulares da FATEC

TIPO DO TRABALHO/ANO: Trabalho de Graduação/2019.

É concedida à FATEC de São José dos Campos: Professor Jessen Vidal permissão para reproduzir cópias deste Trabalho e para emprestar ou vender cópias somente para propósitos acadêmicos e científicos. O autor reserva outros direitos de publicação e nenhuma parte deste Trabalho pode ser reproduzida sem a autorização do autor.

|  |  |
| --- | --- |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Cesar Augusto Siqueira Santos  Rua José Firmino de Moraes, Número 121, Jardim Estoril  12232-020, São José dos Campos – São Paulo |  |

**CESAR AUGUSTO SIQUEIRA SANTOS**

**QuizFATEC: Banco de Questões e Aplicativo Móvel Simulador dos Vestibulares da FATEC**

Trabalho de Graduação apresentado à Faculdade de Tecnologia de São José dos Campos, como parte dos requisitos necessários para a obtenção do título de Tecnólogo em Banco de Dados.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Me. Diogo Branquinho – FATEC SJC**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Me. Eduardo Sakaue – FATEC SJC**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Me. Fernando Masanori – FATEC SJC**

**\_\_\_\_\_/\_\_\_\_\_/\_\_\_\_\_**

**DATA DA APROVAÇÃO**

Dedico este trabalho a todos que me apoiaram no decorrer do mesmo, a minha família por me permitir estudar, a minha noiva por sempre me motivar e buscar sempre o melhor, e ao meu grande amigo Kevin, que sempre incentivou o desenvolvimento desse trabalho.

**AGRADECIMENTOS**

Agradeço primeiramente a Deus, por me capacitar intelectualmente e fisicamente, para o desempenho deste trabalho. Agradeço a todos os professores, desde a educação infantil até a graduação superior, porque eles foram referência para mim, cada um à sua maneira. Um agradecimento especial ao Professor Emanuel Mineda, por me apoiar na escrita deste trabalho. Agradeço também aos professores Fernando Masanori, Diogo Branquinho, Giuliano Bertoti e Eduardo Sakaue, por serem sempre tão solícitos e me ajudarem ao longo do desenvolvimento desse projeto.

Agradeço também ao Lucas Domingos, Kevin Hizatsuki, Gustavo Soares e Jonathan Souza, por me suportarem ao longo desses 3 anos de FATEC, sem o apoio e as risadas deles não teria concluído este curso.

Finalmente agradeço aos meus pais, por me apoiarem e incentivarem meus estudos desde a educação primária até este momento, agradeço a minha noiva Bruna Ignat, por me apoiar a cada momento e me inspirar constantemente.

**RESUMO**

Ingressar numa instituição de ensino superior é o sonho de milhões de brasileiros, entretanto as dificuldades do dia-a-dia, somados ao complicado mercado de trabalho que enfrentamos atualmente, leva milhões de pessoas a postergar esse sonho, tornando a educação superior sempre um sonho futuro. Pensando assim, esse trabalho foi desenvolvido pensando justamente em auxiliar pessoas nos estudos no momento de ingresso da educação superior. Mais especificamente, esse trabalho foi construído com o objetivo de compor um banco de questões provenientes de provas passadas dos vestibulares da FATEC, uma intuição de ensino gratuito e de qualidade. Além da composição do banco de questões, ao longo desse projeto foi desenvolvido um aplicativo para dispositivo móvel, alimentado pelo banco de questões, trazendo mobilidade e agilidade para o estudo de vestibulandos e interessados. Este trabalho abrange tecnologias voltadas para raspagem de dados, desenvolvimento de aplicativos móveis de maneira hibrida, assuntos referentes a infraestrutura e composição de uma API. A raspagem dos vestibulares se mostrou desafiadora e trabalho manual foi necessário, entretanto, o banco de questões se mostrou viável e completamente funcional, permitindo acesso prático e fácil a vestibulandos.

**Palavras-Chave**: Tecnologia; Raspagem de Dados; Graduação; FATEC; Aplicação Móvel; QuizFATEC.

**ABSTRACT**

Entering a college or university education institution is the dream of millions of Brazilians, however the day-to-day difficulties, added to the complicated labor market that we currently face, leads millions of people to postpone this dream, making college education always a future dream. Weighing this way, this work was developed thinking precisely to help people in their studies when entering the third-level education. More specifically, this work was built with the aim of composing a bank of questions from FATEC’s past entrance exams, a college intuition of free and quality teaching. In addition to the composition of the question bank, an application for a mobile device was developed throughout this project, powered by the question bank, bringing mobility and agility to the study of interested students. This work covers technologies aimed at data scraping, development of mobile applications in a hybrid way, issues related to infrastructure and composition of an API. The scraping of the entrance exams proved to be challenging and manual work was necessary, however, the question bank proved to be viable and fully functional, allowing practical and easy access to students.

**Keywords**: Technology; Data Scraping; University graduate; FATEC; Mobile Application; QuizFATEC.

**LISTA DE QUADROS**

[Quadro 1 - Scrapper - Importações e Declarações Globais 33](#_Toc30852453)

[Quadro 2 - Scrapper - Inserção de Dicionários no MongoDB 34](#_Toc30852454)

[Quadro 3 - Scrapper - Raspagem de Gabarito 35](#_Toc30852455)

[Quadro 4 - Scrapper - PDF\_TO\_TEXT 36](#_Toc30852456)

[Quadro 5 - Scrapper - Text to JSON Question 39](#_Toc30852457)

[Quadro 6 - Scrapper Main Function 42](#_Toc30852458)

[Quadro 7 – Back-end: Conexão com Banco de Dados 43](#_Toc30852459)

[Quadro 8 – Back-end: 3º e 4º Rotas 44](#_Toc30852460)

[Quadro 9 – Back-end: 5º e 6º Rota 45](#_Toc30852461)

[Quadro 10 - Classe DataService 51](#_Toc30852462)

**LISTA DE FIGURAS**

[Figura 1 - Exemplo aplicado de KNN 20](#_Toc30859244)

[Figura 2 - Exemplo de Prova Gerada pelo Super Professor 23](#_Toc30859245)

[Figura 3 - Estrutura básica JSON 27](#_Toc30859246)

[Figura 4 - Exemplo de JSON estruturado 27](#_Toc30859247)

[Figura 5 - Esquema tático aplicação Cordova 28](#_Toc30859248)

[Figura 6 - Arquitetura Geral do Projeto 30](#_Toc30859249)

[Figura 7 - Composição Geral de um Notebook Jupyter 32](#_Toc30859250)

[Figura 8 - Comparativo de Gabaritos 35](#_Toc30859251)

[Figura 9 - Exemplo de Documento Inserido 43](#_Toc30859252)

[Figura 10 - Filtro de Ausentes e Inválidos no Schema do MongoDB 49](#_Toc30859253)

[Figura 11 - Arquitetura do Aplicativo QuizFATEC 50](#_Toc30859254)

[Figura 12 - Tela de Login Preenchida 53](#_Toc30859255)

[Figura 13 - Tela Home 54](#_Toc30859256)

[Figura 14 - Questão de Química 55](#_Toc30859257)

[Figura 15 – Botão Validar 56](#_Toc30859258)

[Figura 16 - Diagrama Reportar Questão 57](#_Toc30859259)

**LISTA** **DE TABELAS**

[Tabela 1 - Vantagens e Desvantagens - Super Professor 22](#_Toc30852738)

[Tabela 2 - Vantagens e Desvantagens – Só Exercícios 23](#_Toc30852739)

[Tabela 3 - Vantagens de Desvantagens - Perguntados 24](#_Toc30852740)

[Tabela 4 - Vantagens e Desvantagens - Aplicativo Detran-SP 24](#_Toc30852741)

[Tabela 5 - Bibliotecas Python 31](#_Toc30852742)

**LISTA DE ABREVIATURAS E SIGLAS**

PDF *Portable Document Format*

API *Application Programming Interface*

FATEC Faculdade de Tecnologia

HTTP Hypertext Transfer Protocol

JSON *JavaScript Object Notation*

SDK *System Development Kit*

TI Tecnologia da Informação

UI *User Interface*

UX *User Experience*

URL *Uniform Resource Locator*

IO *Input Output*

HTML *Hypertext Markup Language*

NoSQL *Not Only Struct Query Language*

IP *Internet Protocol Address*

WSGI *Web Server Gateway Interface*

CORS *Cross-Origin Resource Sharing*

OCR *Optical Character Recognition*

KNN K Vizinhos Mais Próximos

ETL *Extraction Transformation Loading*

**SUMÁRIO**

[1. INTRODUÇÃO 15](#_Toc31048452)

[1.1. Motivação 16](#_Toc31048453)

[1.2 Objetivo 16](#_Toc31048454)

[1.3. Escopo 16](#_Toc31048455)

[2. FUNDAMENTAÇÃO TEÓRICA 18](#_Toc31048456)

[2.1. Vestibulares da FATEC 18](#_Toc31048457)

[2.1. Vestibulares e o Fator Psicológico 18](#_Toc31048458)

[2.2. Técnicas de Obtenção de Dados 18](#_Toc31048459)

[2.2.1. Transcrição Manual 18](#_Toc31048460)

[2.2.2. OCR – Reconhecimento Óptico de Caracteres 19](#_Toc31048461)

[2.2.3. Data Scraping e Python 20](#_Toc31048462)

[2.3. *Extraction – Transformation – Load (ETL)* 21](#_Toc31048463)

[2.4. Processamento de Linguagem Natural e Busca de 22](#_Toc31048464)

[2.5. Tecnologias Semelhantes 22](#_Toc31048465)

[2.5.1. Super Professor – Banco de Questões 22](#_Toc31048466)

[2.5.2. Só Exercícios -Banco de Questões 24](#_Toc31048467)

[2.5.3. Aplicativo Perguntados 1 e 2 24](#_Toc31048468)

[2.5.4. Simulado Detran-SP 25](#_Toc31048469)

[2.6. Jupyter Notebook 26](#_Toc31048470)

[2.7. Banco de Dados NoSQL 26](#_Toc31048471)

[2.7.1. JSON 27](#_Toc31048472)

[2.7.2. MongoDB 28](#_Toc31048473)

[2.8. WebView para Dispositivos Móveis – Ionic e Cordova Framework 28](#_Toc31048474)

[3. DESENVOLVIMENTO 30](#_Toc31048475)

[3.1. Arquitetura Global do Sistema 30](#_Toc31048476)

[3.2 Scrapper 31](#_Toc31048477)

[3.2.1 Scrapper – Importações e Declarações Globais 32](#_Toc31048478)

[3.2.2 Scrapper – Inserção de Dicionários no MongoDB 33](#_Toc31048479)

[3.2.3 Scrapper – Raspagem do Gabarito 34](#_Toc31048480)

[3.2.4 Scrapper – Retirada de Texto do PDF de Prova 36](#_Toc31048481)

[3.2.5 Scrapper – Busca de Textos Inválidos em Questões 38](#_Toc31048482)

[3.2.6 Scrapper – Retirada das Questões do Texto da Prova 38](#_Toc31048483)

[3.2.7 Scrapper – Função Principal 42](#_Toc31048484)

[3.3 Persistência dos dados através de MongoDB 42](#_Toc31048485)

[3.4 *Back-end* em Flask 43](#_Toc31048486)

[3.5 DataService - Comunicação com API 47](#_Toc31048487)

[4. RESULTADOS 49](#_Toc31048488)

[4.1. Raspagem das Questões 49](#_Toc31048489)

[4.2. API desenvolvida 50](#_Toc31048490)

[4.3. Aplicativo Consumidor da API e Banco de Dados 51](#_Toc31048491)

[4.3.2 Telas principais e suas funções 52](#_Toc31048492)

[4.4. Tecnologias semelhantes 56](#_Toc31048493)

[5. CONSIDERAÇÕES FINAIS 57](#_Toc31048494)

[5.1. Uso do Scrapper 57](#_Toc31048495)

[5.2. Tecnologias Aplicadas 57](#_Toc31048496)

[5.3. Contribuições 58](#_Toc31048497)

[5.4. Trabalhos Futuros 58](#_Toc31048498)

[REFERÊNCIAS 59](#_Toc31048499)

# 

# 1. INTRODUÇÃO

Em virtude da crise política e econômica vivenciada no Brasil nos anos de 2017 e 2018, o desemprego cresceu, mais precisamente 3,6% considerando o desemprego em janeiro de 2017 e o mesmo período no ano anterior. (g1 2017, p. 1)

O acirramento da competição no mercado de trabalho, leva milhões de pessoas a buscar atualizações, principalmente no âmbito de cursos superiores e pós-graduações. Pensando nisso, a FATEC foi criada pelo Centro Paula Souza, uma instituição pública, mantida pelo governo do Estado de São Paulo.

A FATEC disponibiliza duas vezes por ano mais 14 mil vagas, distribuídos entre 74 cursos focados na área de tecnologia. Levando milhares de jovens, adultos e até idosos a provarem seus conhecimentos em um vestibular, para se aptos, ingressar na Faculdade de Tecnologia do Estado de São Paulo.

O Centro Paula Souza, órgão responsável pela administração das FATECs e ETECs espalhadas pelo Estado de São Paulo, foi criado em 1969, atualmente vinculado à Secretaria de Desenvolvimento do Estado de São Paulo, o objetivo do Centro Paula Souza sempre foi implantar um ensino técnico voltado para formação de profissionais direcionados ao mercado de trabalho, transpassando as instituições de ensino superior focados em discussões acadêmicas (GOVERNO DO ESTADO DE SÃO PAULO, 2009).

As primeiras FATECs foram implantadas na década de 1970, começando em Sorocaba, interior do estado de São Paulo, e São Paulo capital. Atualmente já passam das 70 unidades, espalhadas em 66 cidades. (FATEC, 2018).

O vestibular da FATEC tem um diferencial em relação aos outros, ele acontece duas vezes por ano, com provas totalmente distintas. Por se tratar de tecnólogos, são cursos de período relativamente curtos e atraem pessoas das mais distintas idades e classes sociais.

Muitos vestibulandos da FATEC, concluíram o ensino médio a décadas e não voltaram a estudar, tornando o vestibular um desafio muito grande. Ainda mais se o vestibulando não conhece o modelo da prova, áreas de conhecimento abrangidas, moldes da redação etc. Por fim, as chances desse vestibulando ingressar na FATEC, ou qualquer vestibular, são pouco animadoras. Será preciso dedicação e estudo, execução de provas anteriores como treino, dedicação especial as matérias que têm mais peso para o curso almejado.

Portanto, a falta de conhecimento das áreas de conhecimento que mais precisam de atenção, estudo e prática antes de se realizar um vestibular, pode custar a vaga de um vestibulando em busca de melhores condições de vida, ou mesmo de uma quinada na carreira profissional.

Muitos vestibulandos precisam prestar mais de uma vez o vestibular, uma para identificar onde é necessária mais atenção, compreender a prova e suas estratégias, e outra para ter chances reais de ingressar na instituição de ensino superior. Isso é um empecilho na vida de quem está desempregado, buscando se especializar para ter uma chance maior no mercado de trabalho, ou até mesmo de um recém-formado no ensino médio buscando continuar seus estudos com qualidade.

Existe, portanto, um abismo claro, entre o desejo de ingressar em uma instituição de ensino superior e a obtenção dos conhecimentos necessários para tal. Atualmente isso tem sido suprido com cursinhos e escolas preparatórias para vestibular, entretanto a maior parte dessas instituições é paga, e a maioria das pessoas que desejam aprimorar seus conhecimentos não despende de tempo ou dinheiro sobressalente para gastos em cursinhos ou escolas preparatórias. Evidenciando ainda mais, a dificuldade de jovens e adultos com carreira estabelecida ingressarem em instituições de ensino superior.

## 1.1. Motivação

Não existir um banco de questões de vestibulares de acesso público e facilitado, agilizando o estudo de vestibulandos. Atualmente as questões estão disponíveis através de arquivos PDF, dificultando a leitura e estudo de vestibulandos.

## 1.2 Objetivo

Aplicação de tecnologias com o objetivo de alimentar um banco de dados, apenas por questões, repostas e textos provenientes de vestibulares, aliado a um aplicativo para dispositivos moveis que permita que vestibulandos revejam questões de vestibulares passados.

## 1.3. Escopo

De maneira geral, a melhor maneira de se preparar para qualquer vestibular, é entender quais os principais assuntos tratados no vestibular em questão e realizar simulações cronometradas para se habituar com o estilo da prova. Segundo Tavorá (2016), professor da LFG Concursos, realizar o exame simulado nas condições exatas da realidade, ajudam na aprovação, devido a familiaridade com o modelo de vestibular.

Realizar um vestibular, com conteúdo e estratégia, mais do que só conhecimento técnico, mas já ter vivência no teste, garantem resultados melhores. Os pontos e a nota são baseados no que foi feito e não no que poderia ser feito. (TASIFANATO, 2018)

Considerando os pontos trazidos assim, o escopo do projeto é o desenvolvimento de um banco de questões alimentado exclusivamente por questões de vestibulares, com um foco principal na FATEC, além disso a criação de um aplicativo para smartphones, com a capacidade de simular provas no modelo dos Vestibulares da FATEC, assim o aluno poderá avaliar seus conhecimentos perante as questões presentes no vestibular alvo.

Quanto ao desenvolvimento do projeto, será feito em 4 etapas:

1 – Raspagem das questões e devidas respostas de vestibulares passados da FATEC.

2 – Alimentação de um banco de dados com as questões e repostas extraídas.

3 – Construção de uma API conectando a aplicação WebView e banco de dados.

4 – Desenvolvimento de um aplicativo para ambas as plataformas moveis (iOS e Android).

A fonte da informação e questões será o próprio site de vestibular da FATEC, que disponibilizam de maneira aberta e gratuita, todas as provas dos vestibulares ocorridos, assim como as o gabarito de respostas e questões anuladas.

A partir da extração dos dados, um banco de dados será alimentado com as questões e soluções. A definição do modelo, modelagem das tabelas e definição dos campos, será feito de maneira a melhorar a performance do sistema.

O aplicativo será feito pensando na possibilidade de multiplataformas, além disso a capacidade de ágil desenvolvimento. O desenvolvimento com linguagem nativa, não se torna obrigatório, porque a aplicação fará pouco uso dos recursos nativos do smartphone. A maioria dos recursos e ações será executada dentro da própria aplicação, como responder simulados, obter questões randômicas e assim por diante.

# 2. FUNDAMENTAÇÃO TEÓRICA

Este capítulo expõe as principais áreas de conhecimento e tecnologias abrangidas durante esse trabalho: vestibulares e fator psicológico, técnicas de raspagem de dados, tecnologias semelhantes, bancos de dados não relacionais e WebView para dispositivos móveis.

## 2.1. Vestibulares da FATEC

A FATEC é uma instituição de ensino tecnológico público, fundada na década de 70, é vinculado ao grupo Centro Paula Souza, atualmente disponibiliza mais de 70 cursos em 66 cidades do estado de São Paulo, atraindo milhares de jovens, adultos e até idosos, que desejam uma formação de nível superior, para que isso seja possível, estes devem passar pelo desafio do vestibular, nos modelos da FATEC, são 54 questões de 10 temas diferentes, além de um tema de redação, que pode ser trabalhado através de um texto dissertativo argumentativo ou uma narrativa. A figura 1 ilustra uma página retirada da segunda edição do vestibular da FATEC no ano de 2019, as questões de número 4 e 5 exibidas na figura pertencem ao tema multidisciplinar. Nos vestibulares da FATEC é recorrente a existência de figuras, charges, textos e tabelas, complicando a extração automática dos textos.

Figura 1 - Vestibular da FATEC 2ª 2019

Uma imagem contendo captura de tela, texto

Descrição gerada automaticamente

[Fonte: FATEC (2020)](http://fatecsjc-prd.azurewebsites.net/moodle/pluginfile.php/65111/mod_resource/content/0/Exemplo_de_Projeto_Android_de_Tempo_Real_com_Firebase.pdf)

## 2.2. Vestibulares e o Fator Psicológico

Cada vez mais populares, os vestibulares são a principal entrada para cursos técnicos, superiores e até para bolsas de estudo. Pensando no ponto de vista micros sociais, o jovem e sua família, sofrem uma angústia ou ansiedade no período que antecede os vestibulares, entretanto o vestibular é a única maneira de ingressar em inúmeras universidades e faculdades, como a FATEC por exemplo. Torna-se então uma etapa de todo estudante ou jovem que deseja continuar os estudos e muitos deles não se sentem preparados para realizar vestibulares, mais precisamente 60% dos jovens (D’VILA, 2003). Questionados a respeito das categorias que se assemelhavam ao seu estado de espírito durante a execução do vestibular, cerca de 20,1% se sentem emocionalmente frágeis, outros 16,6% se sentem fisicamente frágeis e outro grupo, 20,1% se sentem despreparados em relação aos estudos (D’VILA, 2003). Pensando assim, estar preparado para executar um vestibular é de sua importância e uma das principais maneiras de se preparar é a ambientação com o exame e saber as áreas de conhecimento abrangidas pelo vestibular.

## 2.3. Técnicas de Obtenção de Dados

Existem inúmeras formas de se extrair dados de textos, arquivos, imagens, sites etc. Este tópico abrangera algumas das alternativas existentes para raspagem de arquivos.

### 2.3.1. Transcrição Manual

A alternativa mais simples seria a transcrição dos vestibulares passados em documentos para serem inseridos dentro do banco de dados, é menos eficiente e extremamente trabalhosa, levaria até mais tempo do que outros métodos, entretanto até a invenção da imprensa, a produção de livros e documentos era feito de através de um laborioso processo manual, eram utilizados formas e procedimentos padrão. (BEZERRA, 2011). Por mais que as tecnologias avancem ainda existem problemas que o trabalho manual se faz necessário, muito vezes é mais fácil solucionar um problema de maneira manual, dessa maneira é possível garantir integralmente a obtenção dos dados.

### 2.3.2. OCR – Reconhecimento Óptico de Caracteres

*Optical Character Recognition* (OCR), reconhecimento ótico de caracteres em tradução literal, patenteado em 1929 na Alemanha e 1935 nos Estados Unidos, trata-se de algoritmos, ou inteligência artificial para realizar o reconhecimento de caracteres de imagens. As técnicas de OCR tem como essência replicar a capacidade humana de interpretação de símbolos e leitura. Para que essa ação seja possível é necessário aplicar inúmeras tecnologias, porque existem desafios como a variação de fontes, estilos e tamanhos, assim como matérias de impressão, qualidade da imagem, quando o assunto é materiais manuscritos o desafio se torna ainda maior.

As redes neurais são a aplicação de inteligência artificial mais difundida para o OCR, isso se deve a facilidade com o aprendizado, modificando a resposta de acordo com um estímulo de entrada ou aprendizado, essa capacidade se assemelha ao cérebro humano porque 1- o conhecimento é adquirido e interpretado através de um processo de aprendizagem, 2 – o conhecimento adquirido é armazenado através da força entre neurônios, ou pesos sinápticos. (HAYKIN, 2001).

Uma outra maneira técnica utilizado para OCR é o algoritmo de K-vizinho mais próximos, ou KNN, uma técnica de aprendizado baseada em casos, esse algoritmo é fundamentado no princípio de que casos semelhantes deverão ter a mesma classe (PASSOS, 2015). Através de fórmulas matemáticas como a distância euclidiana definida por , ou pela distância de Manhatan, conforme , onde e representam os valores do atributo i nos casos **a** e **b** (PASSOS, 2015). A figura 1 abaixo, exemplifica um caso em que KNN foi utilizado para rotular a estrela, assim sendo um K = 7 (distância) e rótulos A e B representados por pequenos círculos. Dada a distância, a circunferência pontilhada foi desenhada, dentro da circunferência restaram 4 elementos de rotulo A e 3 elementos de rotulo B, assim sendo a estrela foi classificada como rotulo A.

Figura 2 - Exemplo aplicado de KNN

Uma imagem contendo objeto

Descrição gerada automaticamente

[Fonte: PACHECO (2019)](http://fatecsjc-prd.azurewebsites.net/moodle/pluginfile.php/65111/mod_resource/content/0/Exemplo_de_Projeto_Android_de_Tempo_Real_com_Firebase.pdf)

Já existem plataformas online que fazem uso de OCR para certos serviços, como o OnlineCR, trata-se de uma plataforma online gratuita que se utiliza de técnicas de OCR, convertendo documentos PDF digitalizados, fotografias, mídias digitais, faxes em documentos editáveis como documentos .DOC, HTML ou arquivos de textos simples. (ONLINEOCR, 2020).

### 2.3.3. Data Scraping e Python

Com o crescimento da internet, na última década, muita informação está à disposição para aqueles que souberem como buscar, extrair e transformar em essa informação em algo realmente útil, a esse processo é dado o nome de Web Scraping, ou Raspagem Web.

O uso mais comum do Web Scraping é a extração de conteúdos web, principalmente de preços de produtos, utilizados em plataformas de comparadores de preço para e-commerce. Não obstante, muitos jornalistas têm feito uso da raspagem de dados para extrair mais informações, de maneira a corroborar com seus textos e reportagens (ADRIOLO 2009).

Data Scraping, ou Raspagem de Dados, é a ciência de extração dos dados de um determinado ambiente, e inserção em outro ambiente, geralmente existe um processo de tratamento dos dados, tornando-os simples e maleáveis (ADRIOLO, 2009). Nesta acepção, torna-se necessário reconhecer o padrão da fonte dos dados, e através de um script de programação ou software, a raspagem se torna possível. Python é uma das linguagens que mais se destaca na raspagem de dados. Existem inúmeras bibliotecas para raspagem de dados, como BeautifulSoup, PDFMiner, PDFQuery, PyPDF2, entre outas.

#### 2.3.3.1. PDFMiner

O PDFMiner é uma biblioteca de extração de informação de documentos PDF. Diferente de outras ferramentas semelhantes, é focada inteiramente na raspagem e análise de dados em texto. PDFMiner permite obter a localização exata do texto em uma página, assim como outras informações como fontes ou linhas. Além disso, é possível converter o PDF em um formato de HTML, por exemplo, facilitando a interpretação em raspagem de dados (PDFMINER, 2017).

#### 2.3.3.2. PyPDF2

O PyPDF2 é uma caixa de ferramentas totalmente feita em Python, surgiu a partir do projeto pyPDF em 2005, focado na manipulação de documentos, recorte de páginas, criptografia e decriptografia de documentos. PyPDF2, foi lançado em 2011 com o objetivo de ler todos os tipos de PDF, no entanto o projeto se manteve apenas nos arquivos PDF de texto, podendo criar arquivos PDF novos e raspar dados de arquivos PDF existentes (PYPDF, 2019).

## 2.4. *Extraction – Transformation – Load (ETL)*

Extração, transformação e carga, em tradução literal. Trata-se de um procedimento padrão de integração de dados, mais especificamente refere-se à capacidade de transcrição dos dados de uma origem para outro destino. Mais utilizado para construção de *data warehouse*, ou armazém de dados, em tradução literal. O processo de ETL trata da capacidade de extrair dados de um sistema-fonte, em seguida os dados são tratados ou convertidos em um formato condizente com análise que será feita, e finalmente não carregados, ou armazenados, em um outro sistema.

Os processos de ETL se popularizaram nos anos 70, época em que as organizações começaram a utilizar vários repositórios, banco de dados e afins. Fez-se necessário uma maneira de integrar esses dados pulverizados, os procedimentos de ETL se popularizaram. Com o passar dos anos, além da popularização dos *data warehouse*, princípios de ETL e ferramentas de *business intelligence*, o tema acesso aos dados self-service se tornou uma tendência, a transformação de dados nas mãos de qualquer usuário ou profissionais não necessariamente técnicos, esse tipo de abordagem tem aumentado a agilidade organizacional, liberando o setor de TI de difundir os dados de diferentes formatos entre os funcionários. O aumento de produtividade é palpável, difundindo mais dados e direcionando as decisões baseadas em dados (SAS, 2020).

Python tem sido difundido como uma ferramenta útil de ETL também, o uso de Python para trabalhos de ETL é recomendado em 3 casos principais: 1 – O Desenvolvedor se sente confortável com o desenvolvimento em python de maneira a compor sua própria ferramenta de ETL; 2 – Trata-se de um caso extremamente simples de ETL; 3 – Trata-se de um caso extremamente especifico, de maneira que apenas através um código customizado, o processo de ETL pode ser feito. (PARKER, 2019)

## 2.5. Processamento de Linguagem Natural e Busca de

/\*TODO: Falar de processamento de linguagem natural aplicada a identificar textos.

## 2.6. Tecnologias Semelhantes

O principal objetivo desse projeto de graduação foi permitir que pessoas possam concentrar seus estudos para o vestibular de maneira mais fácil, independentemente do local em que estejam, para isso a alimentação de um banco de dados se fez necessário, permitindo a mobilidade e agilidade nos estudos. Existiram aplicativos que foram usados como referências para o desenvolvimento do projeto, ambos fazem uso de bancos de dados de questões e são maneiras de transparecer esses bancos de questões. Durante a pesquisa desse projeto, foram encontrados alguns exemplos e banco de questões disponibilizados através de uma plataforma com custo mensal. A seguir, uma breve descrição das tecnologias semelhantes.

### 2.6.1. Super Professor – Banco de Questões

Super Professor é uma plataforma Web que disponibiliza mais de 150 mil questões de diferentes assuntos, cerca de 5 mil questões preparatórias para o ENEM (SPRWEB, 2020), a plataforma disponibiliza ainda dois planos de consumo, são eles:

* Plano Escola: disponibiliza para todos os professores, número de usuários ilimitados e acessos simultâneos ilimitados, da escola a ferramenta para elaboração de avaliações de maneira mais rápida e com mais qualidade. Além disso, este plano concede acesso por IP específico além de espaço em disco para armazenamento de arquivos (provas) gerados. O custo é preço é obtido sob consulta.
* Plano Professor: disponibiliza apenas um acesso, não é permitido acesso simultâneo, e contém apenas 250mb para armazenamento de arquivos. Os custos variam de R$127,00 até R$836,00.

Além disso a plataforma contém um acesso exclusivo para alunos que podem acessar simulados lista de exercícios, consultar resultados através de relatórios e gráficos.

Figura 3 - Exemplo de Prova Gerada pelo Super Professor

Uma imagem contendo texto, livro

Descrição gerada automaticamente

Fonte: SPRWEB (Adaptado, 2019)

Na figura 2 é possível ver uma prova gerada pela plataforma Super Professor, são questões extraídas de vestibulares como UFMG, ITA, FUVEST etc. As questões contêm o ano em que estiveram nos vestibulares, além de possuir figuras, textos e fontes (SPRWEB, 2019). A tabela 1 descreve um balanço entre as vantagens e desvantagens da plataforma Super Professor.

Tabela 1 - Vantagens e Desvantagens - Super Professor

|  |  |
| --- | --- |
| Vantagens | Desvantagens |
| Plataforma repleta de exercícios e de variados assuntos. | Poucas questões focadas em vestibulares. |
| Diferentes modelos de pagamento, com planos para escolas e docentes. | Não possui questões da FATEC. |
| Criação de provas, com questões de vestibulares anteriores. |  |

Fonte: Autor (2019)

Até a publicação deste documento, a plataforma Super Professor ainda não continha questões do Vestibular da FATEC.

### 2.6.2. Só Exercícios -Banco de Questões

De semelhante modo, Só Exercícios é um banco de questões que contém cerca de 14 mil questões, oferece simulados dinâmicos, com questões aleatórias extraídas de provas anteriores do vestibular em foco; busca de questões, permitindo pesquisa por vestibular, ano, disciplinas e assuntos específicos; estatísticas de desempenho, através de gráficos e relatórios; módulos de estudo, através de algoritmos e frequência analisa quais assuntos foram foco nas últimas edições de determinado vestibular. (SOEXERCICIOS, 2020). A tabela 2 descreve as principais vantagens e desvantagens desse banco de questões.

Tabela 2 - Vantagens e Desvantagens – Só Exercícios

|  |  |
| --- | --- |
| Vantagens | Desvantagens |
| Uma plataforma mais completa, contém módulos de estudos dedicados. | Não possui questões de vestibulares da FATEC. |
| Oferecem simulados dinâmicos. | Não oferece um simulado pronto para o vestibular da FATEC. |
| Apresenta estáticas de desempenho, ótimo para manter o aluno focado nos pontos de melhoria. |  |
| Módulos de estudos customizados, com foco maior nos vestibulares. |  |

Fonte: Autor (2019)

Até a publicação deste documento, a plataforma Só Exercícios ainda não continha questões do Vestibular da FATEC.

### 2.6.3. Aplicativo Perguntados 1 e 2

Perguntados é um jogo para dispositivos móveis, lançado em outubro de 2013, foi desenvolvido pela Etermax, está disponível para Android, iOS e Aplicação para Facebook (PERGUNTADOS, 2019). O objetivo do jogo é conquistar seis personagens da roleta, esses personagens representam categorias das perguntas, são elas: artes, ciências, esportes, entretenimento, geografia e história. O primeiro que conquistar os 6 personagens ganha a partida, cada partida possui até 25 rodadas (PERGUNTADOS, 2019).

Esse aplicativo serviu como uma referência de interface e descontração, um aplicativo que remete a estudos dificilmente cai no gosto das pessoas, mas esse aplicativo se tornou bem popular ao longo dos anos, Perguntados 1 já passou da casa dos 100.000.000 de downloads, enquanto Perguntados 2 possui mais de 10.000.000 de downloads (PLAY STORE, 2019). Além disso, a interface amigável e descontraída com certeza aperfeiçoou a experiencia do cliente, e serviu de inspiração para o desenvolvimento desse projeto. A tabela 3 trata sobre as vantagens e desvantagens do aplicativo Perguntados.

Tabela 3 - Vantagens de Desvantagens - Perguntados

|  |  |
| --- | --- |
| Vantagens | Desvantagens |
| Interface amigável e minimalista. | Questões de temas diversos, não existe especificidade. |
| Incentiva, através de um jogo, os estudos. | Não abrange temas tratados regularmente em vestibulares. |

Fonte: Autor (2019)

### 2.6.4. Simulado Detran-SP

O Simulado do Detran-SP, foi uma inspiração funcional do projeto, desenvolvido com um propósito específico, auxiliar estudantes do curso teórico em busca de sua CNH. O curso teórico obrigatório para se obter a CNH é composto por 45 horas de aula, em seguida o estudante deve aguardar duas semanas até realizar a prova teórica oficial, nesse período o aluno é encorajado a continuar estudando em casa e esse aplicativo de auxiliado muitas pessoas, o aplicativo já conta com mais de um milhão de downloads (PLAY STORE, 2019).

A prova teórica do Detran é composta por 30 questões de múltipla escolha que devem ser respondidas em 40 minutos. O banco de dados do Detran, possui cerca de 600 questões de acesso público e transparente a todas as pessoas que se interessarem, semelhante ao caso da FATEC, que também disponibiliza vestibulares anteriores junto com os respectivos gabaritos, porém, apenas através de arquivos PDF, não existe um banco de dados especifico com uma API aberta.

O aplicativo Simulado do Detran-SP serviu principalmente como referência de proposito, cumprindo o papel de auxiliar pessoas interessadas em aprender mais e se acostumar com o modelo do exame teórico. A tabela 4 trata sobre as vantagens e desvantagens do aplicativo, para dispositivos móveis, Simulado Detran-SP.

Tabela 4 - Vantagens e Desvantagens - Aplicativo Detran-SP

|  |  |
| --- | --- |
| Vantagens | Desvantagens |
| Plataforma com tema específico. | Interface pouco amigável. |
| Tem auxiliado estudantes no processo de emissão de Carteira Nacional de Habilitação. | Questões fixas, sem atualização recorrente. |
|  | Disponível apenas para o estado de São Paulo. |

Fonte: Autor (2019)

## 2.7. Jupyter Notebook

Jupyter Notebook, é uma aplicação web e *open-source*, aberta para padrões e serviços interativos computacionais para uma gama de linguagens de programação (JUPYTER NOTEBOOK, 2019). A plataforma do Jupyter é amplamente utilizada para limpeza e tratamento de dados, assim como manipulação de grandes *dataframes* de dados, sua manipulação simplificada, é possível através das células individuais, além da facilidade de documentação e comentários. Os scripts criados no Jupyter são facilmente lidos e manuseados através da sua construção modular.

## 2.8. Banco de Dados NoSQL

Os bancos de dados relacionais são predominantes no mercado, mas com o passar dos anos e avanço da tecnologia e a interação humana com os sistemas computadorizados se fez necessário evoluir e agregar outras maneiras de se armazenar os dados, com isso surgiram armazenamento de dados orientados a objetos e XML. O fato é que com o passar dos anos o conceito de um banco de dados genérico de proposito variável multifacetado se torna insustentável (NOSQL, 2019). Através dessa abertura, o movimento NoSQL começou a tomar força, com a intenção inicial de criar um banco de dados moderno e escalável para aplicações Web (NOSQL, 2019). O termo NoSQL vem do inglês “Not Only SQL”, em tradução literal, não somente SQL. Podendo ainda ser descrito como “NoACID” (FORBES, 2010).

### 2.8.1. JSON

JSON é um dado intercambiável de armazenamento leve, é lido naturalmente por humanos e facilmente convertido e interpretado por máquinas (JSON, 2019). Um objeto JSON é comumente formado por um conjunto chave-valor, mas pode possuir um conjunto de chave-valor estruturado numa lista, formando um array, principal estrutura para armazenar os dados dentro do Firebase.

Figura 4 - Estrutura básica JSON



[Fonte: JSON (2019)](http://fatecsjc-prd.azurewebsites.net/moodle/pluginfile.php/65111/mod_resource/content/0/Exemplo_de_Projeto_Android_de_Tempo_Real_com_Firebase.pdf)

Na imagem 4, consta um exemplo de JSON estruturado num conjunto de chave-valor inserido numa lista contendo outros conjuntos de chave-valor, os textos em azul são as chaves, como “Questão01”, “Questão”, “(A)”, “(B)”, “(C)”, “(D)”, “(E)”, em tom cobre estão os valores das respectivas chaves, importante atentar ao fato de “Questão01” equivale a um novo conjunto de chaves-valor, sendo possível então umas estrutura de estruturas de chave-valor.

Figura 5 - Exemplo de JSON estruturado



Fonte: Autor (2019)

A maior vantagem do JSON em relação aos arquivos XML, é a facilidade de leitura, sendo possível uma leitura fácil por humanos e por máquinas (JSON, 2019).

### 2.8.2. MongoDB

O MongoDB é um banco de dados de propósito geral, baseado em documentos, é um banco de dados distribuído, trata-se de um banco de dados para era moderna de soluções em nuvem (MONGODB, 2019). Foi desenvolvido em Python e possuía uma sinergia facilitada com esta linguagem de programação. É amplamente difundido para fins específicos, por possuir um *schema* flexível, pode ser utilizado de maneira ágil no desenvolvimento de data warehouse, por exemplo. O MongoDB possui ainda uma plataforma na nuvem gratuita para estudantes, através de alguns cliques é possível hospedar um cluster com MongoDB configurado (MONGODB CLOUD, 2020).

## 2.9. WebView para Dispositivos Móveis – Ionic e Cordova Framework

Assim como a tecnologia evolui em torno das redes de computadores e banco de dados, o mesmo ocorreu com o desenvolvimento de páginas para internet e mais recentemente para dispositivos móveis, uma das tecnologias que tem despontado é uso de WebView, consiste em desenvolvimento de páginas para computador e também para dispositivos móveis como tablets e smartphones, isso é possível através de um compilador que converte essa WebView em apk, formato padrão de aplicativos android, instaláveis pela própria loja de aplicativos nativa dos smartphones android.

Ionic Framework é um conjunto de ferramentas open source, criado com o objetivo de desenvolver aplicativos Web e para dispositivos móveis usando tecnologias Web, como HTML, JavaScript e CSS (IONIC, 2019).

O Framework Ionic se mostrou o mais vantajoso, principalmente pela possibilidade de acesso nativo para Android e iOS através de uma única WebView, construída pelo conjunto de ferramentas do Ionic, os aplicativos desenvolvidos usando Ionic são suportados pelo Android, a partir da versão 4.4, e o iOS a partir da versão 10.

O Cordova é um framework open source, capaz de converter a WebView em uma instalável, tanto para dispositivos Android como sistemas iOS, conforme diagrama mostrado na figura 5.

Figura 6 - Esquema tático aplicação Cordova

Uma imagem contendo captura de tela

Descrição gerada automaticamente

Fonte: Adaptado Ionic Fórum (2019)

Considerando a figura 5, mostrada acima, existem dois grandes grupos Aplicação Cordova e Mobile OS. Considerando o grupo alaranjado definido como Aplicação Cordova, temos três grupos menores definidos como:

Web App: consiste no conjunto Web da aplicação, composto por frameworks, linguagens, configurações e estruturas referentes a uma página de site. Aqui temos um site coeso.

Cordova Plugins: consiste na caixa de ferramentas do Cordova, é a camada intermediária entre a WebView e o conjunto de acessórios do celular, tanto de hardware como acelerômetro, câmera, armazenamento, quanto de software como contatos, outros aplicativos. Mecanismo de Renderização HTML: intersecção entre o sistema operacional do dispositivo móvel, APIs HTML e APIs Cordova.

SO (sistema operacional) do dispositivo móvel, em destaque pelo grupo azul, representa o sistema operacional do dispositivo móvel, é o gerenciador de hardware e software do celular.

# 3. DESENVOLVIMENTO

Este capítulo discorrerá a respeito do desenvolvimento do sistema, desde o banco de dados NoSQL até o desenvolvimento da API, permitindo o compartilhamento e livre acesso das questões raspadas. O projeto foi desenvolvido em etapas, conforme:

1 – Raspagem das Provas e Gabaritos com Python

2 – Estruturação e carga do Banco de Dados NoSQL

3 – Desenvolvimento de um API com Flask

4 – Desenvolvimento do APP com WebView

## 3.1. Arquitetura Global do Sistema

A figura 6 ilustra a arquitetura geral da Aplicação trabalhada durante esse projeto de graduação.

Figura 7 - Arquitetura Geral do ProjetoUma imagem contendo captura de tela

Descrição gerada automaticamente

Fonte: Autor (2019)

O desenvolvimento do projeto se dará através de um processo de ETL orientado com python, o início reside na extração das questões e respostas disponibilizadas em arquivos PDFs, pelo próprio Centro Paula Souza, isso será feito através de um Script Python usando PDFMiner e PyPDF2, assistidos pelo Jupyter Notebook. Depois de raspar as questões e repostas, os dados serão carregados em um banco de dados NoSQL MongoDB, numa estrutura de acesso em tempo real, através de uma solução cluster em nuvem. Com o banco estruturado em arquitetura de árvore contendo o texto da questão, alternativas e reposta, um aplicativo será desenvolvido usando a estrutura de WebView com o *framework* Ionic, através de Angular, CSS e HTML 5. A codificação, lógica e aplicação de serviços e métodos será feito através de TypeScript, dessa maneira o projeto será multiplataforma, com desenvolvimento simultâneo de Web Site e também aplicação de Smartphones (IONIC, 2019). A compilação da aplicação será feita através do Apache Cordova Framework.

## 3.2 Scrapper

Scrapper, ou raspador em tradução literal, é script python construído para a finalidade de extrair os textos, converter em um formato de dicionário e carregar no destino MongoDB, o banco de questões, conforme um processo de ETL. Este script foi redigido no Jupyter Notebook.

Figura 8 - Composição Geral de um Notebook Jupyter

Uma imagem contendo captura de tela

Descrição gerada automaticamente

Fonte: Autor (2019)

A única necessidade prévia para utilização dos Scrapper é que os arquivos referentes a prova e gabarito, precisam estar no mesmo diretório físico que o notebook jupyter Scrapper.

Os arquivos, formato .ipynb utilizados no Jupyter são chamados de Notebooks, bloco de notas em tradução literal (Autor, 2019). Ao longo dessa seção serão esclarecidos os blocos de código do Scrapper.

### 3.2.1 Scrapper – Importações e Declarações Globais

A primeira célula do Scrapper trata das importações de bibliotecas utilizadas ao longo do script, são elas:

Tabela 5 - Bibliotecas Python

|  |  |  |
| --- | --- | --- |
| Biblioteca | Versão | Função |
| pymongo[srv] | 3.9.0 | Biblioteca responsável pela administração da conexão do código python com o MongoDB atlas cluster. |
| PDFMiner | 2014032018 | Dessa biblioteca são importadas várias ferramentas para um conjunto de ações como iterações no PDF, navegação de páginas, pdfpage, responsável pela extração do texto de cada página etc. |
| io | 3.7.5rc1 | Biblioteca padrão do Python, de acesso a dispositivos de IO, entrada e saída, utilizado para acessar a função StringIO que permite a leitura em fluxo na memória para um texto. |
| PyPDF2 | 1.26.0 | Biblioteca redigida para facilitar a leitura de PDF através do Python, como teve alguns problemas de codificação, foi utilizada apenas para acessar o número de páginas de cada prova e para raspar os gabaritos, onde a codificação e formatação não apresentaram problemas. |

Fonte: Autor (2019)

Todas os *imports* necessários e as versões utilizadas estão disponíveis no arquivo requeriements.txt, são facilmente instaláveis através do comando *pip*, gerenciador de pacotes do python (PIP, 2019).

Quadro 1 - Scrapper - Importações e Declarações Globais

|  |  |
| --- | --- |
| 1 | import pymongo |
| 2 | from pdfminer.pdfinterp import PDFResourceManager, PDFPageInterpreter |
| 3 | from pdfminer.pdfpage import PDFPage |
| 4 | from pdfminer.converter import TextConverter |
| 5 | from pdfminer.layout import LAParams |
| 6 | from io import StringIO |
| 7 | from PyPDF2 import PdfFileReader |
| 8 |  |
| 9 | prova = "PROVA\_2\_2018" |
| 10 | qtdInseridas = 0 |
| 11 | gabarito = prova.replace("PROVA", "GABARITO") |
| 12 | temas = ['MULTIDISCIPLINAR','RACIOCÍNIO LÓGICO','HISTÓRIA','QUÍMICA','INGLÊS','MATEMÁTICA','FÍSICA','GEOGRAFIA', 'BIOLOGIA','PORTUGUÊS','REDAÇÃO','RASCUNHO', 'RAC', 'LÓGICO'] |
| 13 | respgabarito = {} |
|  |  |

Fonte: Autor (2019)

Essa célula também instancia alguns objetos globais como a variável que carrega a edição da prova que será raspada pelo script, assim como a edição do gabarito correspondente. O vetor temas foi criado para limpar os textos referentes aos temas das provas e gabaritos, esses temas são mais facilmente acessados através do número da questão, que segue o mesmo padrão desde a primeira edição da prova da FATEC. O vetor “respgabarito” armazena as questões globalmente de maneira que possam ser acessadas dentro de diferentes células funções.

### 3.2.2 Scrapper – Inserção de Dicionários no MongoDB

Essa célula é dedicada a função de inserção no banco e dados NoSQL, nesse projeto foi utilizado o MongoDB, desenvolvido em Python, facilitando a implementação conjunta com o Scrapper. A conexão é criada através da biblioteca pymongo, utilizando da função MongoClient, que permite instanciar uma interface de acesso aos recursos do banco de dados alocado no cluster em nuvem do Atlas Cluster, dentro dessa interface é possível acessar o banco QuizFATEC e a *collection* provas, onde todas as questões ficam armazenadas. O quadro 2 retoma o código python da função de inserção do dicionário no banco de dados.

Quadro 2 - Scrapper - Inserção de Dicionários no MongoDB

|  |  |
| --- | --- |
| 1 | def insert\_question(json): |
| 2 | global qtdInseridas |
| 3 | qtdInseridas += 1 |
| 4 | client = pymongo.MongoClient("mongodb+srv://admin:admin@quizfatec-xl7tb.mongodb.net/test?retryWrites=true&w=majority") |
| 5 | dbProvas = client["QuizFATEC"] |
| 6 | colProvas = dbProvas["provas"] |
| 7 | try: |
| 8 | colProvas.update\_one({"\_id" : json['\_id']}, {"$set": json}, upsert=True) |
| 9 | print("Questão "+json['\_id']+" Inserida com Sucesso") |
| 10 | except: |
| 11 | print("Não foi possível inserir a questão") |
| 12 | client.close() |
| 13 | def insert\_question(json): |
|  |  |

Fonte: Autor (2019)

A função recebe como parâmetro um dicionário de textos montado pela função *text\_to\_json\_question*, entretanto a variável “colProvas” que administra a *collection* do MongoDB interpreta esse dicionário de *strings* do python como um JSON estruturado. A exceção no momento da inserção pode ser acionada por motivos como falha na conexão com o cluster na nuvem devido à falta de conexão ou IP bloqueado, um outro motivo relevante é que no MongoDB todos os objetos inseridos nas *collections* possuem um \_id gerado automaticamente, mas que podem ser alterados, como é o caso nesse projeto, onde os \_id de cada objeto foram alterados de maneira a possuírem um padrão referente a cada prova e edição, facilitando que a leitura futura por uma aplicação cliente, como é o caso do aplicativo.

### 3.2.3 Scrapper – Raspagem do Gabarito

Os gabaritos possuíam um padrão estabelecido, a figura 8 ilustra o comparativo entre os gabaritos do primeiro semestre de 2010 e segundo semestre de 2019, uma diferença de quase uma década entre edições mas que não refletiu em mudanças significativas de padrão.

Figura 9 - Comparativo de Gabaritos



Fonte: Autor (2019)

Conforme a figura 8, é possível identificar que a estrutura do gabarito se manteve inalterada ao longo dos vestibulares. Atributos como estrutura bi colunar, número de questões padrão de 54, ordem dos temas e gabarito similar facilitaram a raspagem permitindo que a função responsável pela raspagem do gabarito fosse mais simples e 100% efetiva para todos os vestibulares, o quadro 3 abrange a função *awnser\_extractor* mencionada.

Quadro 3 - Scrapper - Raspagem de Gabarito

|  |  |
| --- | --- |
| 1 | def answer\_extractor(path): |
| 2 | arrumarPadrao = "QUESTÃO ALTERNATIVADISCIPLINA QUESTÃOALTERNATIVA DISCIPLINA" |
| 3 | padrao = "QUESTÃO ALTERNATIVA DISCIPLINA QUESTÃO ALTERNATIVA DISCIPLINA" |
| 4 |  |
| 5 | with open(path, 'rb') as f: |
| 6 | gabarito=PdfFileReader(f) |
| 7 | text = gabarito.getPage(gabarito.numPages-1).extractText().upper().replace('\n','').replace(' ',' ').replace(arrumarPadrao,padrao) |
| 8 |  |
| 9 | #tratamento de exceção na prova 2 de 2019 |
| 10 | if prova == 'PROVA\_2\_2019': |
| 11 | text = text.replace('34 MATEMÁTICA', '34 ANULADA MATEMÁTICA') |
| 12 | for tema in temas: |
| 13 | text = text.replace(tema,'') |
| 14 |  |
| 15 | lstSemCabecalho = text[text.find(padrao)+len(padrao):].split(' ') |
| 16 |  |
| 17 | questao = [] |
| 18 | resposta = [] |
| 19 | cont = 0 |
| 20 |  |
| 21 | for el in lstSemCabecalho: |
| 22 | if el!='' and cont%2==0: |
| 23 | questao.append(el) |
| 24 | cont+=1 |
| 25 | elif el!='' and cont%2 ==1: |
| 26 | resposta.append(el) |
| 27 | cont+=1 |
| 28 |  |
| 29 | for i in range(0,54): |
| 30 | respgabarito.update({int(questao[i]):resposta[i]}) |
|  |  |

Fonte: Autor (2019)

A função *answer\_extractor* utiliza do PyPDF2 para transformar o arquivo PDF num texto único excluindo imagens do cabeçalho e transpondo a tabela para um texto simples tabulado, através da função padrão de *strings* do python, *replace* é possível retirar textos desnecessários como os temas das questões, espaços múltiplos e as linhas puladas ao longo do texto. Todas os gabaritos da FATEC possuem um texto padrão composto pela sequência de palavras: “QUESTÃO ALTERNATIVA DISCIPLINA QUESTÃO ALTERNATIVA DISCIPLINA”, apresentado repetido devido as duas colunas usadas na composição do gabarito, conforme ilustrado na figura 9. Essa sequência de palavras repetidas permitiu a leitura consistente da resposta correta para cada questão excluindo o cabeçalho e as disciplinas mantendo apenas os numerais e respectivas respostas de cada questão. Finalmente, o dicionário de respostas foi composto utilizando o número da questão como chave, permitindo seu acesso para pesquisa ao longo da raspagem das questões.

### 3.2.4 Scrapper – Retirada de Texto do PDF de Prova

A função *pdf\_to\_text* descrita abaixo, no quadro 4, recebe como parâmetro um endereço de diretório onde se encontra a prova e a edição que será raspada.

Quadro 4 - Scrapper - PDF\_TO\_TEXT

|  |  |
| --- | --- |
| 1 | def pdf\_to\_text(pdfname): |
| 2 | rsrcmgr = PDFResourceManager() |
| 3 | sio = StringIO() |
| 4 | device = TextConverter(rsrcmgr, sio, codec='utf-8', laparams=LAParams()) |
| 5 | interpreter = PDFPageInterpreter(rsrcmgr, device) |
| 6 | with open(pdfname, 'rb') as fp: |
| 7 | for page in PDFPage.get\_pages(fp): |
| 8 | interpreter.process\_page(page) |
| 9 | text = sio.getvalue() |
| 10 | qtdPages = PdfFileReader(fp) |
| 11 | #Remove os temas, são mais facilmente encontrados pelo número das questões |
| 12 | for tema in temas: |
| 13 | text = text.replace(tema,'') |
| 14 | #Tira o espaçamento de linhas, facilitando o tratamento do texto |
| 15 | text = text.replace("\n","") |
| 16 | #Tratamento de ocorrências de nomenclatura do vestibular |
| 17 | for i in range(1, qtdPages.numPages): |
| 18 | text = text.replace(str(i)+"VESTIBULAR "+pdfname[13]+"o SEM/"+pdfname[15:19]+" • FATEC ","") |
| 19 | text = text.replace(str(i)+" VESTIBULAR "+pdfname[13]+"o SEM/"+pdfname[15:19]+" • FATEC ", "") |
| 20 | text = text.replace("VESTIBULAR "+pdfname[13]+"o SEM/1"+pdfname[15:19]+" • FATEC "+str(i),"") |
| 21 | text = text.replace("VESTIBULAR "+pdfname[13]+"o SEM/"+pdfname[15:19]+" • FATEC "+str(i),"") |
| 22 | #Cria as chaves de Identificação de Questão |
| 23 | for i in range(0,54): |
| 24 | if i < 10: |
| 25 | n = 'Questão0' + str(i) |
| 26 | ni= '0'+str(i)+'Questão' |
| 27 | else: |
| 28 | n = 'Questão' + str(i) |
| 29 | ni = str(i) + 'Questão' |
| 30 | text = text.replace(n, '[-Chave-]'+n).replace(ni,'[-Chave-]'+n).replace('Leia o texto ', '[-Chave-]Leia o texto ').replace('Leia os textos ', '[-Chave-]Leia os textos ') |
| 31 | #Cria as chaves de Identificação das Alternativas |
| 32 | text = text.replace('(A)', '[-ChaveA-](A)').replace('(B)', '[-ChaveA-](B)').replace('(C)', '[-ChaveA-](C)').replace('(D)', '[-ChaveA-](D)').replace('(E)', '[-ChaveA-](E)') |
| 33 | fp.close() |
| 34 | device.close() |
| 35 | sio.close() |
| 36 | return text |
|  |  |

Fonte: Autor (2019)

Essa função objetiva interpretar os PDFs e concatená-los em um bloco de texto de maneira a facilitar toda e qualquer manipulação, para isso foi utilizado a biblioteca PDFMiner, através do PDFResourceManager e PDFPageInterpreter foi possível manipular o PDF interagindo ao longo das páginas e permitindo alguns tratamentos que se seguiram. Uma vez que a variável text recebeu o texto concatenado de todas as páginas os seguintes tratamentos ocorreram, em primeiro momento foram retirados os textos dos temas, semelhante ao tratamento aplicado na raspagem de gabaritos, ocorrências como “MATEMÁTICA”, “RACICIONIO LÓGICO”, “QUÍMICA”, “REDAÇÃO” etc. foram removidos do bloco de texto. Em segundo as quebras de linhas foram removidas, facilitando o armazenamentos dos textos assim como sua manipulação, em terceiro momento os textos padrão de rodapé são removidos, todas as páginas das provas contém uma identificação referente a cada edição do vestibular e número da página, um laço de repetição itera da primeira até a última página buscando no texto e removendo-o.

Como dito anteriormente, todas os vestibulares da FATEC contém 54 questões, dessa maneira um laço de repetição substitui todas as ocorrências de identificação de questão para uma chave que será usada para quebrar o texto em um vetor, facilitando a iteração, na prática todas as ocorrências do texto Questão somando ao numeral identificador da questão são substituídos por uma *tag* que será usada na quebra do texto para vetor. A *tag* utilizada foi criada de maneira a garantir sua não ocorrência em nenhum vestibular, dessa maneira a chave de questões foi definida como [-Chave-].

Finalmente acontece o processo de substituição das alternativas identificadas pelo caractere do alfabeto de A até E, sempre cercadas por parêntesis, permitindo sua identificação, objetivando a facilidade da interpretação também foi inserida uma *tag* única construída seguindo o padrão da *tag* usada nas questões. Ao final do código é retornado o bloco de texto concatenado e tratado.

### 3.2.5 Scrapper – Busca de Textos Inválidos em Questões

Considerando a inviabilidade na raspagem de textos referenciados, imagens, ilustrações e charges esta função foi criada, essa função retorna um a resposta booleana baseado na ocorrência de conjuntos semânticos que retomem textos que o próprio script não pode raspar. Essa marcação será utilizada para separar as questões que precisarão de um trabalho manual e individual.

Quadro 5 - Scrapper – Find Text Image in Question

|  |  |
| --- | --- |
| 1 | def find\_text\_image\_in\_question(textQuestion): |
| 2 | invalidar = ["de acordo com o texto", "de acordo com a figura", "de acordo com a imagem", |
| 3 | "segundo o texto", "segundo a figura", "segundo a imagem", "a figura acima", |
| 4 | "a figura abaixo", "a imagem acima", "a imagem a baixo", "o texto acima", |
| 5 | "o texto abaixo", "charge"] |
| 6 | for word in invalidar: |
| 7 | if word.upper() in textQuestion.upper(): |
| 8 | return False |
| 9 | else: |
| 10 | return True |
|  |  |

Fonte: Autor (2019)

Trata-se de uma função python, de complexidade N. Primeiramente um vetor é instanciado contendo conjuntos de palavras que fazem alusão a referencias que o script não é capaz de interpretar de maneira automática, para formulação desse vetor, foi feito um levantamento com base nas edições do vestibular da FATEC que foram alvo da raspagem. Em seguida um laço de repetição é utilizado para percorrer cada conjunto semântico do vetor, durante esse processo, a presença dos conjuntos semânticos é verificada dentro do texto da questão, passado como parâmetro.

### 3.2.6 Scrapper – Retirada das Questões do Texto da Prova

Esta função foi desenvolvida com o objetivo de realmente retirar os conjunto de questão e suas alternativas do bloco total do texto, existiu a complexidade de se estabelecer um padrão de raspagem dentro de um vestibular que apresenta em média 24 páginas, 54 questões de 10 temas de conhecimento diferentes, com dezenas de imagens, textos e links para consulta. Embora as provas tenham alguns padrões definidos como a nomenclatura e numerologia padrão, existem outras ocorrências fora de padrão como referências de acesso a links são referenciados como “Acesso em 15.03.2019” ou “Acesso em 17/05/2010” e até mesmo “Acesso em 15.03.09”, outra dificuldade são os textos para múltiplas questões que costumam ser referenciados antes da primeira questão do intervalo mas variam em quantidade, podem ser aplicados para duas ou mais questões, em certas ocorrências existem também imagens.

A variável “valida” presente no código se refere a validade da questão, considerando se a mesma está apta a ser respondida pelo usuário do banco de questões, para isso são considerados os parâmetros: as alternativas foram raspadas corretamente; texto da questão não utiliza como referência imagens, figuras, charges ou textos que não foram identificados pelo Scraper; erro na leitura do texto da questão. Através dessa variável o Scrapper permitirá a identificação das questões que demandam um tratamento manual.

O quadro 6 contém o código de transformação do bloco concatenado em questões numa estrutura de dicionário que pode ser facilmente convertida em um objeto JSON.

Quadro 6 - Scrapper - Text to JSON Question

|  |  |
| --- | --- |
| 1 | def text\_to\_json\_question(textProva): |
| 2 | vetorText = textProva.split('[-Chave-]') |
| 3 |  |
| 4 | for elemento in vetorText: |
| 5 | if elemento[:7] == "Questão" and elemento[7:8] != "": |
| 6 |  |
| 7 | nQuestao = int(elemento[7:9]) |
| 8 |  |
| 9 | textoQst = elemento[9:] |
| 10 |  |
| 11 | arrayQuestoes = elemento[9:].split('[-ChaveA-]') |
| 12 |  |
| 13 | valida = True |
| 14 |  |
| 15 | #Busca de Links na Questão, poderão ser usados na WebView |
| 16 | preLinks = elemento[9:].split('<') |
| 17 | arrayLinks = [] |
| 18 | for el in preLinks: |
| 19 | if el[:4] == "http" and el.find('>') != -1: |
| 20 | arrayLinks.append(""+ el[:el.find('>')] +"") |
| 21 |  |
| 22 | dicionarioQuestao = {} |
| 23 | dicionarioQuestao.update({'\_id': prova + "\_QUESTAO\_" + str(nQuestao)}) |
| 24 | dicionarioQuestao.update({'prova':prova}) |
| 25 | dicionarioQuestao.update({'numero': nQuestao}) |
| 26 |  |
| 27 | if nQuestao >= 1 and nQuestao <= 9: |
| 28 | dicionarioQuestao.update({'tema': 'MULTIDISCIPLINAR'}) |
| 29 | elif nQuestao >= 10 and nQuestao <= 14: |
| 30 | dicionarioQuestao.update({'tema': 'RACIOCÍNIO LÓGICO'}) |
| 31 | elif nQuestao >= 15 and nQuestao <= 19: |
| 32 | dicionarioQuestao.update({'tema': 'HISTÓRIA'}) |
| 33 | elif nQuestao >= 20 and nQuestao <= 24: |
| 34 | dicionarioQuestao.update({'tema': 'QUÍMICA'}) |
| 35 | elif nQuestao >= 25 and nQuestao <= 29: |
| 36 | dicionarioQuestao.update({'tema': 'INGLÊS'}) |
| 37 | elif nQuestao >= 30 and nQuestao <= 34: |
| 38 | dicionarioQuestao.update({'tema': 'MATEMÁTICA'}) |
| 39 | elif nQuestao >= 35 and nQuestao <= 39: |
| 40 | dicionarioQuestao.update({'tema': 'FÍSICA'}) |
| 41 | elif nQuestao >= 40 and nQuestao <= 44: |
| 42 | dicionarioQuestao.update({'tema': 'GEOGRAFIA'}) |
| 43 | elif nQuestao >= 45 and nQuestao <= 49: |
| 44 | dicionarioQuestao.update({'tema': 'BIOLOGIA'}) |
| 45 | elif nQuestao >= 50 and nQuestao <= 54: |
| 46 | dicionarioQuestao.update({'tema': 'PORTUGUÊS'}) |
| 47 | else: |
| 48 | dicionarioQuestao.update({'tema': 'INVÁLIDO'}) |
| 49 | valida = False |
| 50 |  |
| 51 | try: |
| 52 | textoQst = arrayQuestoes[0] |
| 53 | except: |
| 54 | textoQst = 'Falha na Leitura da Questão' |
| 55 | valida = False |
| 56 | dicionarioQuestao.update({'texto': textoQst}) |
| 57 |  |
| 58 | try: |
| 59 | alternativaA = arrayQuestoes[1].replace('(A)','').lstrip().rstrip() |
| 60 | except: |
| 61 | alternativaA = 'Falha na Leitura da Alternativa A' |
| 62 | valida = False |
| 63 | dicionarioQuestao.update({'a': alternativaA}) |
| 64 | [...] |
| 65 | try: |
| 66 | alternativaE = arrayQuestoes[5].replace('(E)','').lstrip().rstrip() |
| 67 | except: |
| 68 | alternativaE = 'Falha na Leitura da Alternativa E' |
| 69 | valida = False |
| 70 | dicionarioQuestao.update({'e': alternativaE}) |
| 71 |  |
| 72 | if len(arrayLinks) > 0: |
| 73 | dicionarioQuestao.update({'links': arrayLinks}) |
| 74 |  |
| 75 | dicionarioQuestao.update({'resposta': respgabarito.get(nQuestao)}) |
| 76 |  |
| 77 | if valida == True: |
| 78 | dicionarioQuestao.update({'valida': find\_text\_image\_in\_question(textoQst)}) |
| 79 | else: |
| 80 | dicionarioQuestao.update({'valida': valida}) |
| 81 |  |
| 82 | if dicionarioQuestao['resposta'] != 'ANULADA': |
| 83 | insert\_question(dicionarioQuestao) |
| 84 |  |

Fonte: Autor (2019)

Logo na primeira linha da função já ocorre uma quebra em vetor, utilizando como chave para o *split* a *tag* [-Chave-]. Em seguida um laço de repetição itera ao longo dos elementos desse vetor cruzando os primeiros caracteres do texto de maneira a validar a ocorrência do conjunto identificador de Questão e o numeral correspondente, caso o elemento corresponda a uma questão válida, a sequência de tratamentos se inicia. Num primeiro momento é obtido o número da questão, que será utilizado na pesquisa das respostas do dicionário de respostas raspadas do gabarito. Em seguida se inicia o processo de raspagem dos links. Os links foram raspados dessa maneira com o objetivo de tratar mais facilmente na aplicação cliente, será possível disponibilizar o link para consulta do próprio usuário da aplicação. Seguindo pelo código, são instanciados alguns elementos do dicionário como questão, prova e o \_id. O MongoDB como descrito anteriormente, é um banco de dados NoSQL de *schema* flexível, onde o foco da performance é a leitura, dessa maneira inserir esses dados em campos no dicionário facilitam a manipulação e pesquisa tanto no banco de dados em si, como na aplicação cliente.

Os temas de conhecimento são inseridos através do número da questão, respeitando o intervalo estabelecido no padrão das edições que se seguiram após 2016. O texto da questão é obtido após a quebra das *tags* “[-ChaveA-]”, o primeiro elemento do vetor é o próprio texto da questão, na sequência as alternativas como seus textos são obtidos dentro de um tratamento de exceção, que caso acionado invalida o texto da alternativa, dessa maneira será possível tratar essa questão individualmente e manualmente no futuro.

Em seguida a lista de links raspadas é inserida no dicionário, nesse momento o *schema* flexível do MongoDB se mostra vantajoso, permitindo que caso a questão não possua links, não será carregado no banco uma lista vazia, consequentemente acarretando redução de armazenamento do banco de dados e agilizando a interpretação na aplicação cliente.

Finalmente a reposta, que foi raspada e armazenada durante a função *awnser\_extractor* é pesquisada através da função *get*, nativa de dicionários python, completando a estrutura do dicionário, ao final do código uma estrutura condicional impede que questões anuladas sejam inseridas no banco de dados.

### 3.2.7 Scrapper – Função Principal

*Main function*, ou aplicação principal, em tradução livre, é a função que o código python interpreta como início da execução, assim sendo, nessa função é estabelecido o *path* composto pela edição do vestibular que será raspado concatenado com o diretório onde o arquivo está salvo. Em seguida a variável *pathGabarito* é instanciada de semelhante modo. O quadro 7 abrange a função principal do Scrapper.

Quadro 7 - Scrapper Main Function

|  |  |
| --- | --- |
| 1 | if \_\_name\_\_ == '\_\_main\_\_': |
| 2 | path ="Provas\\"+ prova + ".pdf" |
| 3 | pathGabarito ="Gabaritos\\"+ gabarito + ".pdf" |
| 4 | answer\_extractor(pathGabarito) |
| 5 | print("Inicio de Leitura") |
| 6 | text\_to\_json\_question(pdf\_to\_text(path)) |
| 7 | print("Fim de Leitura") |
|  |  |

Fonte: Autor (2019)

Após instanciar as variáveis de diretório, a função de raspagem de gabaritos é chamada, armazenando na variável global respgabarito. Uma mensagem de início de leitura foi inserida de maneira a facilmente identificar o início e fim da raspagem dos vestibulares. Em seguida são invocadas as funções de *pdf\_to\_text* e *text\_to\_json\_question*, funções responsáveis pela raspagem dos vestibulares e inserção no banco de dados.

## 3.3 Persistência dos dados através de MongoDB

Uma vez raspados os vestibulares e os gabaritos, a inserção no banco de dados aconteceu através da biblioteca python pymongo, utilizada na função *insert\_question*. Caso o banco de dados escolhido para essa aplicação fosse um banco relacional tradicional, seria necessária uma sequência de tabelas e relacionamentos para se obter uma o simples relacionamento de link e questão, questão e alternativa correta, prova e questões etc. Além disso o MongoDB proporciona automaticamente *dashboards*, conforme figura 9, com métricas calculadas em cima dos dados cadastrados, facilitando uma análise superficial, para identificação de dados ausentes ou qualquer anomalia na inserção dos dados.

O *schema*, embora flexível, foi padronizado seguindo os atributos das questões, na figura 9 é possível ver um documento com todos os dados cadastrados.

Figura 10 - Exemplo de Documento Inserido

Uma imagem contendo captura de tela

Descrição gerada automaticamente

Fonte: Autor (2019)

Os campos “\_id”, “Prova”, “Questao”, “Tema”, “(A)”, “(B)”, “(C)”, “(D)”, “(E)”, Resposta são obrigatórios e devem estar presentes em todas os documentos, o campo de Links aparece apenas nas questões em que links foram raspados, dessa maneira o espaço de armazenamento do banco de dados é otimizado.

## 3.4 *Back-end* em Flask

O *back-end* do projeto foi desenvolvido com Python, seguindo na mesma linguagem do script de rasgagem, para isso a biblioteca *Flask* foi escolhida, para criar a comunicação entre o banco de dados e a WebView. *Flask* é um leve WSGI, um framework de aplicações *web* (FLASK, 2019). Foram mapeadas ao todo 6 rotas, algumas requisições *get* e outras *post*, a figura 8 retrata o código inicial do *back-end*:

Quadro 8 – Back-end: Conexão com Banco de Dados

|  |  |
| --- | --- |
| 1 | from flask import Flask, jsonify, request |
| 2 | from flask\_pymongo import PyMongo |
| 3 | from flask\_cors import CORS |
| 4 | import pymongo |
| 5 | import random |
| 6 |  |
| 7 | app = Flask(\_\_name\_\_) |
| 8 | CORS(app, resources=r'/QuizFATEC/\*') |
| 9 |  |
| 10 | client = pymongo.MongoClient("mongodb+srv://admin:admin@quizfatec-xl7tb.mongodb.net/test?retryWrites=true&w=majority") |
| 11 | dbProvas = client["QuizFATEC"] |
| 12 |  |
| 13 | @app.route('/QuizFATEC/Provas',methods=['GET']) |
| 14 | def get\_all\_questions(): |
| 15 | colProvas = dbProvas["provas"] |
| 16 | output = [] |
| 17 | for q in colProvas.find(): |
| 18 | output.append({'texto': q['texto'], 'a': q['a'], 'b': q['b'], 'c': q['c'], 'd': q['d'], 'e': q['e'], 'reposta': q['resposta'], 'tema': q['tema']}) |
| 19 |  |
| 20 | return jsonify({'resp' : output}) |
| 21 |  |
| 22 | @app.route('/QuizFATEC/Provas/<id>', methods=['GET']) |
| 23 | def get\_one\_question(id): |
| 24 | colProvas = dbProvas["provas"] |
| 25 | q = colProvas.find\_one({'\_id': id}) |
| 26 | if q is None : |
| 27 | output = 'a busca nao retornou resultados' |
| 28 | else: |
| 29 | output = {'texto': q['texto'], 'a': q['a'], 'b': q['b'], 'c': q['c'], 'd': q['d'], 'e': q['e'], 'reposta': q['resposta'], 'tema': q['tema']} |
| 30 |  |
| 31 | return jsonify(output) |
|  |  |

Fonte: Autor (2019)

No quadro 8, as primeiras linhas contêm a importação das bibliotecas necessárias para a execução do mesmo, em seguida uma aplicação *Flask* é instanciada, o CORS é configurado de maneira a receber todas as rotas contendo “/QUIZFATEC/” como parte da rota. Assim é possível fazer o acesso de diferentes interpretados, como Google Chrome, Safari, Mozilla Firefox etc. Com o CORS configurado, o *client* do MongoDB é instanciado, e a *collection* QuizFATEC é selecionada, permitindo a iteração com os registros durante as rotas.

Em seguida se inicia a imposição de rotas através de funções, a primeira rota instanciada é uma requisição *get* que retorna todas as questões cadastradas no banco de dados, essa rota foi usada principalmente para efeito de testes com o Postman, uma plataforma colaborativa para desenvolvimento de APIs (POSTMAN, 2019). A segunda rota mapeada é uma função para retornar apenas uma questão baseada em um ID passado, caso o não exista um documento no banco de dados correspondente a função uma mensagem padrão será retornada.

Quadro 9 – Back-end: 3º e 4º Rotas

|  |  |
| --- | --- |
| 1 | @app.route('/QuizFATEC/Provas/Temas/<tema>', methods=['GET']) |
| 2 | def get\_random\_by\_theme(tema): |
| 3 | colProvas = dbProvas["provas"] |
| 4 | lst = [] |
| 5 | for q in colProvas.find({'tema': tema}): |
| 6 | try: |
| 7 | lst.append({'texto': q['texto'], 'a': q['a'], 'b': q['b'], 'c': q['c'], 'd': q['d'], 'e': q['e'], 'resposta': q['resposta'], 'prova': q['prova'], 'numero':q['numero'], 'tema': q['tema'], 'links': q['links']}) |
| 8 | except: |
| 9 | lst.append({'texto': q['texto'], 'a': q['a'], 'b': q['b'], 'c': q['c'], 'd': q['d'], 'e': q['e'], 'resposta': q['resposta'], 'prova': q['prova'], 'numero':q['numero'], 'tema': q['tema']}) |
| 10 |  |
| 11 | i = random.randint(0, len(lst)) |
| 12 | try: |
| 13 | output = lst[i] |
| 14 | except: |
| 15 | output = "A busca nao retornou resultados" |
| 16 |  |
| 17 | return jsonify(output) |
| 18 |  |
| 19 | @app.route('/QuizFATEC/Provas/Random/', methods=['GET']) |
| 20 |  |
| 21 | def get\_random(): |
| 22 | colProvas = dbProvas["provas"] |
| 23 |  |
| 24 | lst = [] |
| 25 | for q in colProvas.find(): |
| 26 | lst.append({'texto': q['texto'], 'a': q['a'], 'b': q['b'], 'c': q['c'], 'd': q['d'], 'e': q['e'], 'reposta': q['resposta'], 'prova': q['prova'], 'numero':q['numero'], 'tema': q['tema']}) |
| 27 |  |
| 28 | i = random.randint(0, len(lst)) |
| 29 | try: |
| 30 | output = lst[i] |
| 31 | except: |
| 32 | output = "A busca nao retornou resultados" |
| 33 | return jsonify({'resp' : output}) |
|  |  |

Fonte: Autor (2019)

A terceira rota é principal rota da aplicação, trata-se de uma rota para requisições *get* que, é utilizada pela WebView, para obter as questões aleatórias baseado no tema selecionado, a estrutura de exceção é utilizada para tratar as questões que não possuem links raspados pelo Scrapper. A quarta rota foi criada para propósitos empírico, é uma requisição *get* sem parâmetros que retorna um documento aleatório raspada pelo Scrapper, nessa rota os links foram desconsiderados.

Quadro 10 – Back-end: 5º e 6º Rota

|  |  |
| --- | --- |
| 1 | @app.route('/QuizFATEC/Usuarios/', methods=['POST']) |
| 2 | def post\_user(): |
| 3 | colUsuarios = dbProvas["usuarios"] |
| 4 |  |
| 5 | id = request.json['\_id'] |
| 6 |  |
| 7 | colUsuarios.update\_one({"\_id" : id}, {"$set": request.json}, upsert=True) |
| 8 |  |
| 9 | q = colUsuarios.find\_one({'\_id': id}) |
| 10 | if q is None : |
| 11 | output = '' |
| 12 | else: |
| 13 | output = {'\_id': q['\_id'], 'email': q['email'], 'name':q['name'], 'nickname':q['nickname']} |
| 14 |  |
| 15 | return jsonify(output) |
| 16 |  |
| 17 | @app.route('/QuizFATEC/Usuarios/Login', methods=['POST']) |
| 18 | def get\_authenticated(): |
| 19 | colUsuarios = dbProvas["usuarios"] |
| 20 |  |
| 21 | id = request.json['\_id'] |
| 22 | password = request.json['password'] |
| 23 |  |
| 24 | q = colUsuarios.find\_one({'\_id': id}) |
| 25 | if q is None : |
| 26 | output = "Usuario nao cadastrado" |
| 27 | elif q['password'] != password: |
| 28 | output = "Senha invalida, verifique a ortografia" |
| 29 | else: |
| 30 | output = {'\_id': q['\_id'], 'email': q['email'], 'name':q['name'], 'nickname':q['nickname']} |
| 31 |  |
| 32 | return jsonify(output) |
| 33 |  |
| 34 | if \_\_name\_\_ == '\_\_main\_\_': |
| 35 | app.run(debug = True) |
|  |  |

Fonte: Autor (2019)

A quinta rota é uma requisição *post*, criada para cadastrar novos usuários ou atualizar os existentes, isso é possível através da função de *upsert* do MongoDB . Em seguida é feito uma busca do usuário inserido, de maneira a retornar o documento recém cadastrado ou atualizado. A última rota/função é referente a autenticação na aplicação, trata-se de uma requisição POST que recebe dois parâmetros, login e senha. Uma consulta é realizada no banco buscando pela ocorrência do *username*, que é a chave única identificadora dos documentos definidos no banco de dados. Em seguida, acontece a validação da senha, caso seja inválida uma mensagem é retornada, tratando especificamente da senha incorreta, caso seja válida, o objeto é retornado pela função, para que aplicação web consiga interpretar esses dados, através de uma mensagem customizada na tela inicial, por exemplo.

## 3.5 DataService - Comunicação com API

A classe DataService foi desenvolvida com TypeScript, tem o papel de impor as requisições GET e POST utilizadas pela aplicação. O resultado retornado na API é interpretado pelas telas responsáveis pela chamada. O quadro 11 abrange a composição e funções/rotas estabelecidas dentro do Aplicativo.

Quadro 11 - Classe DataService

|  |  |
| --- | --- |
| 1 | import { Injectable } from '@angular/core'; |
| 2 | import { HttpClient, HttpHeaders } from '@angular/common/http'; |
| 3 |  |
| 4 | @Injectable({ |
| 5 | providedIn: 'root' |
| 6 | }) |
| 7 | export class DataService { |
| 8 |  |
| 9 | public baseUrl = "http://bc9394bd.ngrok.io/QuizFATEC"; |
| 10 | //public baseUrl = "http://localhost:5000/QuizFATEC"; |
| 11 |  |
| 12 | constructor(private http: HttpClient) { } |
| 13 |  |
| 14 | public getAuthenticated(data: any) { |
| 15 | return this.http.post(`${this.baseUrl}/Usuarios/Login`, data); |
| 16 | } |
| 17 |  |
| 18 | public postUser(data: any) { |
| 19 | return this.http.post(`${this.baseUrl}/Usuarios/`, data); |
| 20 | } |
| 21 |  |
| 22 | public getAllQuestions() { |
| 23 | return this.http.get(`${this.baseUrl}/Provas`); |
| 24 | } |
| 25 |  |
| 26 | public getRandomQuestion() { |
| 27 | return this.http.get(`${this.baseUrl}/Provas/Random/`); |
| 28 | } |
| 30 |  |
| 31 | public getRandomTheme(theme: string) { |
| 32 | return this.http.get(`${this.baseUrl}/Provas/Temas/${theme}`); |
| 33 | } |
| 34 | } |
|  |  |

Fonte: Autor (2019)

Essa classe é utilizada sempre que a comunicação com a API se faz necessária, dessa maneira, foram estabelecidas rotas para:

* getAuthenticated: função post responsável por enviar login e senha e retornar um documento JSON com os dados do usuário logado, ou uma mensagem de erro é retornada informando que a senha está errada ou usuário não cadastrado.
* postUser: função responsável por enviar os dados para cadastro do usuário, esses dados são enviados via JSON intrínseco na rota POST, ou também efetuar alguma atualização de login, devido a função upsert do MongoDB.
* getAllQuestions: esta rota foi utilizada principalmente para testes no início do desenvolvimento da aplicação, nenhum parâmetro é passado e todas as questões e repostas armazenadas no banco de dados são retornadas. Trata-se de uma requisição GET.
* getRandomQuestion: assim como rota comentada acima, essa função foi utilizada para testes, nenhum parâmetro era passado e uma questão completamente aleatória era retornada. Trata-se de uma requisição GET.
* getRandomTheme: função mais utilizada pela aplicação, a função recebe como parâmetro o texto da área de conhecimento selecionada, dentro das áreas abrangidas pelo Vestibular da FATEC, em seguida um JSON é retornado com uma questão sortida do tema selecionado.

# 4. RESULTADOS

Este capítulo tem o objetivo de pontuar os resultados colhidos com após o desenvolvimento e alimentação do banco de questões e comparar com as tecnologias estudas e aplicações semelhantes levantadas durante o capítulo 2.

## 4.1. Raspagem das Questões

Considerando como motivação principal o desenvolvimento e alimentação de um banco de questões, com foco no vestibular da FATEC, foram raspadas Durante o processo de raspagem e alimentação do banco de dados, foram analisadas as provas de vestibulares ocorridos entre o primeiro semestre de 2016 e o segundo semestre de 2019, totalizando 8 provas, ou 432 questões. Destas questões, a aplicação *Scrapper* identificou 368 questões, resultando em um aproveitamento de 85%.

Considerando as 368 questões identificadas como total, a aplicação encontrou textos vazios, provavelmente por que as alternativas eram compostas por imagens ou fórmulas, em 7% das 368, aproximadamente 26 questões. Cerca de 19 questões, ou 5% das 368 questões, apresentaram algum tipo de falha na leitura das alternativas, o que acabou invalidando a questão. Portanto, considerando o total de 432 questões, das 8 provas analisadas, 368 questões foram identificadas pelo *Scrapper* e 323, ou 75% foram raspadas com sucesso.

/\*TODO: colocar a métrica de resultado do processamento natural + questões marcadas como invalidas\*/

O MongoDB proporciona ainda, um *dashboard*, com uma análise superficial e automática, entretanto, é de grande valia, ajudando a identificar facialmente a taxa de questões inválidas ou com textos ausentes, conforme exibido na figura 10.

Figura 11 - Filtro de Ausentes e Inválidos no Schema do MongoDB

Uma imagem contendo captura de tela

Descrição gerada automaticamente

Fonte: Autor (2020)

Através desse mesmo *dashboard* é possível obter inúmeras informações, por exemplo, as questões que possuem links possuem em média apenas um link, mas variam de 1 a 3. A raspagem mais efetiva foi a primeira edição do vestibular da FATEC no ano de 2016, empatada com a segunda edição do ano de 2017, ambas com 16% das 368 questões raspadas. A raspagem apontou que 26% das respostas eram a letra B e apenas 17% apresentaram a alternativa C como correta.

## 4.2. API desenvolvida

Para fazer o relacionamento entre o banco de dados e qualquer aplicação de maneira segura e transparente, o desenvolvimento de uma API se fez necessário. Nesse projeto foi utilizado uma biblioteca Python muito popular, o Flask, que permite subir uma API com poucas linhas de código. Nesta API, foram implementadas 6 rotas, 4 foram criadas com o objetivo de extrair questões, dessa maneira é possível extrair questões com temas específicos, questões exatas, através da combinação de edição, ano e número. Uma rota retorna questões completamente aleatórias, e uma retorna todas as 368 questões armazenadas no banco de questões.

Esta API, teve o proposito, nesse projeto, de permitir a comunicação entre o banco de dados MongoDB e o aplicativo QuizFATEC, feito em Ionic. Entretanto, para um trabalho futuro, esta API poderia ser aberta de maneira a fomentar outros aplicativos, sites de estudos, ou qualquer outra tecnologia que usasse desses dados para um bem para sociedade.

## 4.3. Aplicativo Consumidor da API e Banco de Dados

O Aplicativo foi desenvolvido utilizando o framework Ionic, com TypeScript para as logicas de programação das interfaces, e Angular, para desenvolvimento das WebView, CSS foi utilizado para caracterizar e customizar alguns elementos. A figura 11 ilustra a arquitetura geral do aplicativo, comunicação entre telas e afins.

Figura 12 - Arquitetura do Aplicativo QuizFATEC

Uma imagem contendo captura de tela

Descrição gerada automaticamente

Fonte: Autor (2019)

Através da figura 11, acima, exemplifica os comportamentos e ações disponíveis dentro do aplicativo, a primeira tela acessada é a tela de Login, onde o usuário pode realizar login ou se cadastrar, através da tela de cadastro. Uma vez cadastrado, o usuário realiza o login e a tela home é disponibilizado, conforme mostrado na figura 11 os blocos verdes são acessíveis apenas com login efetuado. Na tela Home o usuário terá a opção de selecionar o tema de estudos desejados, uma vez selecionado o tema, a aplicação redireciona o usuário a tela respectiva, onde a questão será exibida junto com as respectivas alternativas. O usuário pode selecionar uma das alternativas e validar a reposta, ou reportar uma questão caso ela esteja ilegível segundo critérios do próprio usuário da aplicação, nesse caso um e-mail será gerado automaticamente, no aplicativo de e-mail padrão do smartphone. Caso o usuário reporte a questão ou acerte a reposta correta, ele passa a ter apenas a opção de voltar a tela home, onde pode selecionar novamente um tema e o ciclo se repete.

Pensando na agilidade do desenvolvimento e facilidade na correção de ponto único de falha, foi criado um componente Angular, denominado “*Question*”, que foi utilizado em todas as telas referentes as questões.

### 4.3.2 Telas principais e suas funções

A tela de login é a primeira tela a ser exibida quando o aplicativo é instalado, nessa tela o usuário do sistema tem a possibilidade de realizar o login, após preencher os campos usuário e senha, que são obrigatórios, ou abrir a conta através do botão “ABRA SUA CONTA AQUI!”. A figura 12 exibe a interface inicial da aplicação.

Figura 13 - Tela de Login Preenchida



Fonte: Autor (2019)

A tela de login funciona como barreira para o uso da aplicação apenas por aqueles que tiverem os seus dados pessoais persistidos no ambiente da aplicação. Essa tela possui também validadores de campos obrigatórios, limitador de tamanho de texto para o campo senha.

A tela home é a interface principal da aplicação e deriva para as telas de quiz, nessa tela o usuário tem a capacidade de selecionar o tema para ênfase de estudos. A figura 13 ilustra a tela inicial da aplicação.

Figura 14 - Tela Home

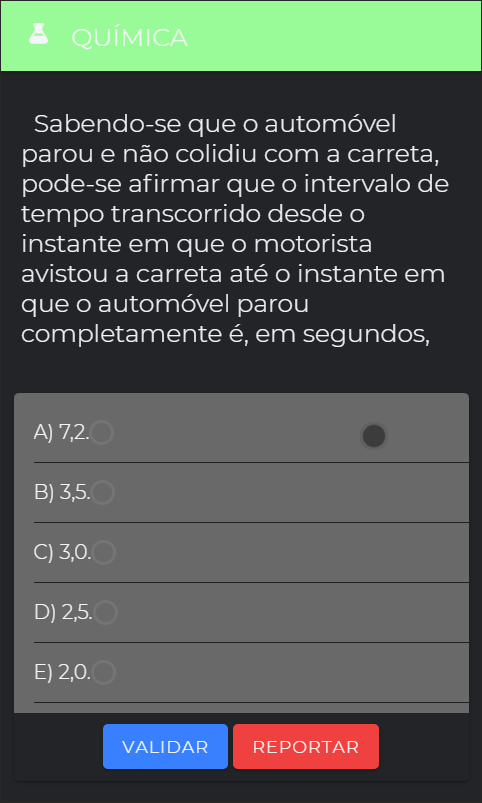


Fonte: Autor (2019)

A figura 13 ilustra ainda uma mensagem boas vindas seguida do apelido do usuário, o objetivo é manter a comunicação leve e prática.

Conforme figura 13, a aplicação possui 10 telas para questões, cada uma é customizada segundo a interface da tela Home, isso foi feito pensando na melhor experiencia do cliente. A figura 14 ilustra a composição da tela, o primeiro texto é sempre referente ao texto das questões, em seguida as alternativas são colocadas, e no fim da tela existem dois botões, “VALIDAR” e “REPORTAR”.

Figura 15 - Questão de Química



Fonte: Autor (2019)

A figura 15 ilustra os retornos possíveis após o do botão “VALIDAR” ter sido acionado. O objetivo é atender ao requisito funcional referente a reposta em tempo real durante o momento do quiz. Caso o usuário do sistema tenha acertado a reposta uma mensagem em tom verde é exibida no fim da tela retornando uma mensagem de sucesso e incentivando os estudos, conforme ilustrado na figura 15, em seguida o botão de validar é substituído pelo botão “VOLTAR AO MENU”, que leva o usuário a tela Home, onde ele pode selecionar novamente uma tema de questão a ser estudado.

Figura 16 – Botão Validar

Uma imagem contendo captura de tela

Descrição gerada automaticamente

Fonte: Autor (2019)

Caso o usuário do sistema erre a resposta, uma mensagem em tom vermelho será exibida na borda inferior da tela junto de uma mensagem estudando o usuário a tentar novamente, o botão de VALIDAR continuará disponível ao usuário até que acerte a reposta.

Finalmente, a tela de questões possui ainda a função de reportar questões, que venham a ser incompreensíveis, considerando que todas as questões foram raspadas através de um script padrão, sem iteração humana, existe uma margem considerável de falha, assim sendo a aplicação foi feita de maneira que caso uma questão esteja inconsistente, o usuário da aplicação poderá reportar a mesma através de um e-mail gerado pela própria aplicação, conforme ilustrado na figura 16.

Figura 17 - Diagrama Reportar Questão

Uma imagem contendo captura de tela, equipamentos eletrônicos

Descrição gerada automaticamente

Fonte: Autor (2019)

A questão de Raciocínio Lógico, ilustrada na figura 16, possui as alternativas em branco, tornando impossível para o usuário responder corretamente à questão, para isso a função reportar foi desenvolvido, através desse botão, o e-mail mostrado na figura é gerado automaticamente no aplicativo de e-mail padrão no smartphone do usuário. O usuário tem a alternativa de enviar o e-mail ou não, a aplicação gera apenas o título, destinatário e texto padrão. O envio do e-mail deve ser feito pelo próprio usuário, ao voltar para o QuizFATEC, tendo enviado o e-mail ou não, os botões de VALIDAR e REPORTAR são ocultados, restando apenas a opção de VOLTAR AO MENU para o usuário.

## 4.4. Tecnologias semelhantes

Foram citadas duas tecnologias semelhantes no capítulo de fundamentação teórica, o aplicativo/jogo Perguntados e o Simulado do Denatran. Comparando o QuizFATEC com essas aplicações é perceptível uma interface mais amigável que o Simulado do Denatran, porém mais simples que o aplicativo Perguntados. Em questão de proposito, o QuizFATEC se assemelha ao Simulado do Denatran, onde o propósito é claro, auxiliar os estudos dos alunos do curso teórico, antes obrigatório, enquanto o Perguntados é um aplicativo de propósito geral, com o objetivo de incentivar uma competição sadia, por conhecimentos de mundo.

# 5. CONSIDERAÇÕES FINAIS

O capítulo que se segue tratará sobre as conclusões obtidas durante e após o desenvolvimento desse projeto e sugestões para trabalhos futuros seguindo esse tema.

## 5.1. Uso do Scrapper

O Scrapper se mostrou uma excelente ferramenta, no processo de ETL especifico de extração dos dados das provas, reduzindo um trabalho manual que tomaria vários dias, para menos de uma hora, isso foi possível devido a substituição da força humana por um script python que carregou cerca de 350 questões de oito provas diferentes num tempo inferior a uma hora. Entanto a raspagem automática mostrou algumas limitações, a impossibilidade de extrair imagens facilmente, grande parte delas contém o link da imagem, isso foi tratado minimizando o impacto da raspagem das imagens. Outra falha mapeada foi a impossibilidade de separar textos válidos para múltiplas questões, sua raspagem se mostrou complexa por falta de padrão estabelecido e por variar muito entre quantidade de questões baseadas no texto. Além desses pontos, existiram ainda algumas questões que por motivos incertos não foram raspadas, provavelmente por falta de identificação clara, presença de palavras variáveis, imagens ou caracteres especiais que não foram identificados automaticamente.

## 5.2. Tecnologias Aplicadas

Após a fundamentação teórica, desenvolvimento e resultados deste trabalho, as seguintes tecnologias e *frameworks* foram aplicadas:

* Raspagem dos dados foi feita utilizando Python, mais propriamente através das bibliotecas PyPDF2 e PDFMiner.
* MongoDB foi o banco de dados utilizado, hospedado na MongoDB *Cloud*, nuvem educacional gratuita própria.
* Para construção da API de comunicação entre o banco de dados e qualquer conexão exterior, Python com o a biblioteca *Flask* foram utilizados.
* O Servidor que hospeda o serviço da API, é uma máquina Ubuntu 16 alocada na nuvem da *Digital Ocean*, o tunelamento é feito através do SSH na porta 8080, gerando um domínio temporário gratuito.
* O Framework Ionic foi utilizado para construir o aplicativo híbrido, fazendo de tecnologias web como HTML, CSS e JavaScript.
* O Aplicativo QuizFATEC foi desenvolvido utilizando, Angular e TypeScript para construção das interfaces do aplicativo, assim como a regras de negócio por trás de cada interface.
* Apache Cordova foi utilizado para compilar o projeto Ionic em nativo para Android.

## 5.3. Contribuições

As contribuições obtidas durante a fundamentação teórica, desenvolvimento e resultados deste trabalho são:

* Através dos estudos aplicados, tornou-se viável a transformação dos arquivos PDF em textos manipuláveis.
* Desenvolvimento de um programa que interpreta os vestibulares da FATEC e modela os dados num formato amigável para manipulação de uma máquina.
* Alimentação de um banco de dados NoSQL, com as questões extraídas e tratadas provenientes dos vestibulares passados da FATEC.
* Construção de um API aberta que permite que livre acesso as questões persistidas no Banco de Questões.
* Desenvolvimento de um aplicativo gratuito, que permite que vestibulandos se familiarizem com o vestibular da FATEC.

## 5.4. Trabalhos Futuros

Visando aprimorar a experiencia e aprendizado do vestibulando que utilizarem o QuizFATEC, as seguintes melhorias foram mapeadas:

* Aprimorar o Scrapper de maneira a categorizar melhor os temas das questões.
* Aprimorar o script de raspagem dos dados maneira que através de processamento linguagem natural, o programa classifique as questões que forem dependentes de temas.
* Inserir a função no aplicativo de selecionar mais questões do mesmo tema, sem necessariamente voltar a tela inicial.
* Inserir a função no aplicativo de simulado, gerando uma prova automática de 54 questões na mesma estrutura dos Vestibulares da FATEC.
* Cadastrar os acertos e erros dos usuários do aplicativo para gerar métricas, possibilitando um sistema de ranking dos usuários.

# REFERÊNCIAS

CRUZ, H. R. **Carona Solidária: Um Aplicativo para Promover Sustentabilidade, Colaboração e Economia na FATEC São José dos Campos.** 65 f. Dissertação (Trabalho de Graduação em Tecnologia de Banco de Dados) - FATEC – Faculdade de Tecnologia Prof. Jessen Vidal, São José dos Campos, 2016.

FORBES, D., GETTING REAL ABOUT NOSQL AND THE SQL ISN’T SCALABLE LIE, Publicação Blog, 2010.

D’AVILA, G. T. **Vestibular: Fatores Geradores de Ansiedade na “Cena da Prova”.** 12 f. Artigo Científico, Universidade Federal de Santa Catarina, Florianópolis, 2003.

O'Reilly Media, Inc. Christopher J. Date. **What is Database Design, Anyway?.** UK n ISBN: 9781492048428. 1 jan. 2016.

G1 SP. **Inscrições para Vestibular da FATEC Aberta**. Disponível em https://g1.globo.com/sp/sao-paulo/noticia/fatec-abre-inscricoes-para-vestibular-de-meio-de-ano-nesta-terca-feira.ghtml Acesso em: 16/09/2019.

ABRIL GUIA DO ESTUDANTE. **Técnicas para Estudo para qualquer prova**. Disponível em https://guiadoestudante.abril.com.br/enem/7-otimas-tecnicas-de-estudo-para-qualquer-prova/ Acesso em: 16/09/2019.

ABRIL GUIA DO ESTUDANTE. **Porque fazer simulados ajuda na hora da prova**. Disponível em https://guiadoestudante.abril.com.br/estudo/por-que-fazer-simulados-ajuda-na-hora-da-prova/ Acesso em: 16/09/2019.

ADMINISTRADORES. **Importância de se preparar par o mercado de trabalho**. Disponível em http://www.administradores.com.br/artigos/carreira/a-importancia-de-preparar-se-para-o-mercado-de-trabalho/109009/ Acesso em 16/09/2019.

PORTAL DO GOVERNO DO ESTADO DE SÃO PAULO. **História do Centro Paula Souza**. Disponível em http://www.saopaulo.sp.gov.br/spnoticias/ultimas-noticias/especial-40-anos-do-centro-paula-souza-conheca-a-linha-do-tempo-da-instituicao/ Acesso em 16/09/2019.

VESTIBULAR DA FATEC. **Unidades e Cursos da FATEC**. Disponível em http://www.vestibularfatec.com.br/unidades-cursos/ Acesso em 20/04/2019.

WIEDERHOLD, G. **The Structural Model for Database Design.** 22 f. Artigo Científico Stanford University, California, 1983.

INFOQ. **Google Firebase: Back-end completo para aplicações web e mobile**. Disponível em https://www.infoq.com/br/news/2016/07/google-firebase Acesso em 13/06/2019.

KNIGHT CENTER FOR JOURNALISM IN THE AMERICAS. **Unraveling data scraping: Understanding how to scrape data can facilitate journalists' work**. Disponível em https://knightcenter.utexas.edu/en/blog/00-9676-unraveling-data-scraping-understanding-how-scrape-data-can-facilitate-journalists-work Acesso em 23/09/2019.

PDFQUERY. **Concise, friendly PDF scraping using jQuery or XPath syntax**. Disponível em https://github.com/jcushman/pdfquery Acesso em 15/05/2019.

PYPDF2. **PyPDF2 Documentation**. Disponível em https://pythonhosted.org/PyPDF2/ Acesso em 17/05/2019.

PYPDF2’S ORIGIN. **Home page for the PyPDF2 project**. Disponível em http://mstamy2.github.io/PyPDF2/ Acesso em 17/05/2019.

APACHE CORDOVA. **Mobile apps with HTML, CSS & JS Target multiple platforms with one code base Free and open source**. Disponível em https://cordova.apache.org/ Acesso em 20/05/2019.

JUPYTER NOTEBOOK. **The Jupyter Notebook**. Disponível em https://jupyter.org/ Acesso em 20/09/2019.

MONGODB. **The MongoDB 4.2 Manual**. Disponível em https://docs.mongodb.com/manual/?\_ga=2.118473677.532038213.1558491340-1615551956.1558491340 Acesso em 08/10/2019.

TINYURL. **Making over a billion long URLs usable! Serving billions of redirects per month**. Disponível em https://tinyurl.com/ Acesso em 08/10/2019.

ORG NOSQL. **Your Ultimate Guide to the Non-Relational Universe!**. Disponível em <http://nosql-database.org/> Acesso em 08/10/2019.

JSON. **Introducing JSON**. Disponível em https://www.json.org/ Acesso em 08/10/2019.

PDFMINER. **pdfminer Release 0.0.1.** Disponível em https://buildmedia.readthedocs.org/media/pdf/pdfminer-docs/latest/pdfminer-docs.pdf Acesso em 10/10/2019.

POSTMAN, **Postman | The Collaboration Platform for API Development.** Disponível em https://www.getpostman.com/ Acesso em 18/10/2019.

PERGUNTADOS. **Perguntados Divirta-se desafiando seus amigos e inimigos no jogo de trivia do momento!.** Disponível em http://www.preguntados.com/pt# Acesso em 18/10/2019.

GOOGLE PLAY. **Perguntados.** Disponível em https://play.google.com/store/apps/details?id=com.etermax.preguntados.lite&hl=pt\_BR Acesso em 18/10/2019.

GOOGLE PLAY. **Perguntados 2**. Disponível em https://play.google.com/store/apps/details?id=com.etermax.trivia.preguntados2&hl=pt\_BR Acesso em 18/10/2019.

GOOGLE PLAY. **Simulado Detran-SP**. Disponível em https://play.google.com/store/apps/details?id=br.gov.sp.detran.simulado&hl=pt\_BR Acesso em 18/10/2019.

SUPER PROFESSOR. **Super Professor Avaliações no Clique do seu Mouse**. Disponível em https://www.sprweb.com.br/mod\_superpro/index.php Acesso em 03/01/2019.

SÓ EXERCICIOS. **Só Exercício O Ensino que se adapta a você**. Disponível em https://soexercicios.com.br/ Acesso em 03/01/2020.

PASSOS, U. B. C.; MATIAS, I. O.; ANDRADE, M.; PASSOS, C. E. S. O. Um Estudo Comparativo Entre Técnicas de Inteligência Computacional para o Reconhecimento Ótico de Caracteres Manuscritos. **XLVII SBPO – Simpósio Brasileiro de Pesquisa Operaciona**l. Ago, 2015.

COMPUTAÇÃO INTELIGENTE. **K vizinhos mais próximos – KNN**. Disponível em http://computacaointeligente.com.br/algoritmos/k-vizinhos-mais-proximos/ Acesso em 05/01/2020.

HAYKIN, S. **Redes Neurais: Princípios e Prática 2.ed**. Porto Alegre: Bookman, 2001.

ONLINE OCR. **Sobre o FREE OCR Online Serviço**. Disponível em https://www.onlineocr.net/pt/service/about Acesso em 25/01/2020.

PARKER, E. **Python & ETL 2019: A List and Comparison of the Top Python ETL Tools**. Disponível em https://www.xplenty.com/blog/python-etl-2019-a-list-and-comparison-of-the-top-python-etl-tools/ Acesso em 26/01/2020.

SAS. **ETL O que é e qual sua importância?** Disponível em https://www.sas.com/pt\_br/insights/data-management/o-que-e-etl.html Acesso em 26/01/2020.

MONGODB CLOUD. **If you can build it, you can build it on MongoDB Cloud.** Disponível em https://www.mongodb.com/cloud Acesso em 26/01/2020.

VESTIBULAR FATEC. **Provas e Gabaritos.** Disponível em https://www.vestibularfatec.com.br/provas-gabaritos/ Acesso em 27/01/2020.