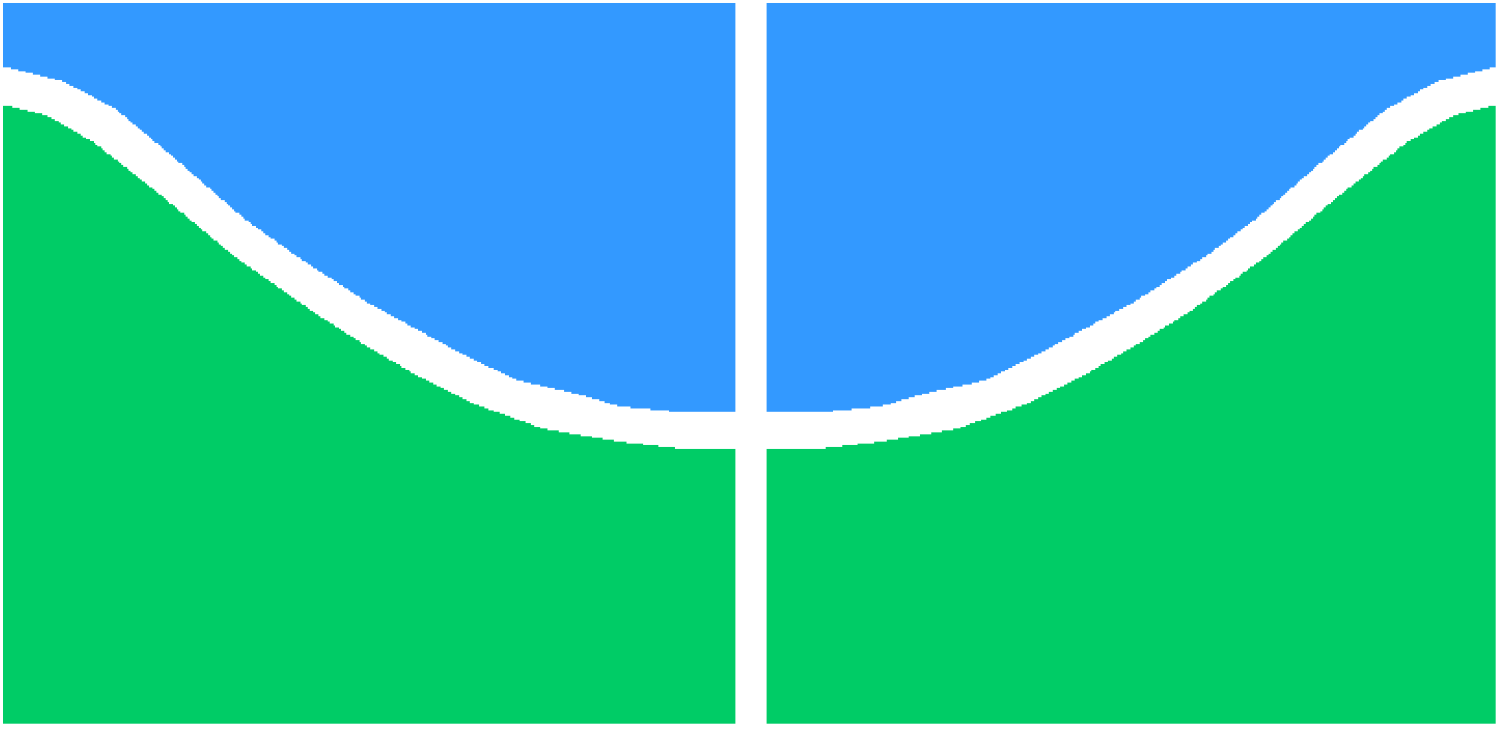


Universidade de Brasília – UnB
Faculdade UnB Gama – FGA
Engenharia de Software

# Guia para o Desenvolvedor Deslocado: um ebook pragmatico para elucidar os principais conceitos de Engenharia de Software

Autores: Guilherme de Lyra Pereira
Max Henrique Barbosa
Orientador: Profa. Dra. Carla Silva Rocha Aguiar

Brasília, DF
2022

Guilherme de Lyra Pereira
Max Henrique Barbosa

# Guia para o Desenvolvedor Deslocado: um ebook pragmatico para elucidar os principais conceitos de Engenharia de Software

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Universidade de Brasília – UnB
Faculdade UnB Gama – FGA

Orientador: Profa. Dra. Carla Silva Rocha Aguiar

Brasília, DF
2022

Guilherme de Lyra Pereira
Max Henrique Barbosa

Guia para o Desenvolvedor Deslocado: um ebook pragmatico para elucidar os principais conceitos de Engenharia de Software/ Guilherme de Lyra Pereira
Max Henrique Barbosa
. – Brasília, DF, 2022-
53 p. : il. (algumas color.) ; 30 cm.

Orientador: Profa. Dra. Carla Silva Rocha Aguiar

Trabalho de Conclusão de Curso – Universidade de Brasília – UnB
Faculdade UnB Gama – FGA, 2022.

1. . 2. . I. Profa. Dra. Carla Silva Rocha Aguiar. II. Universidade de Brasília – UnB. III. Faculdade UnB Gama – FGA. IV. Guia para o Desenvolvedor Deslocado: um ebook pragmatico para elucidar os principais conceitos de Engenharia de Software

CDU 02:141:005.6

Guilherme de Lyra Pereira
Max Henrique Barbosa

# Guia para o Desenvolvedor Deslocado: um ebook pragmatico para elucidar os principais conceitos de Engenharia de Software

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Trabalho aprovado. Brasília, DF, :

---

**Profa. Dra. Carla Silva Rocha Aguiar**
Orientador

---

Convidado 1

---

Convidado 2

Brasília, DF
2022

# Agradecimentos

Gostaríamos de agradecer, primeiramente, aos nossos respectivos familiares: pais e irmãos, que incentivaram-nos nos momentos difíceis dando força para prosseguirmos.

Agradecemos, também, aos nossos professores, pelas correções e ensinamentos que permitiram-nos apresentar um melhor desempenho no nosso processo de formação profissional ao longo do curso.

Por fim, agradecemos às pessoas com as quais convivemos ao longo desses anos de curso, que certamente tiveram impacto em nossa formação acadêmica.

# Resumo

A indústria de software ainda está amadurecendo. Existem inúmeros frameworks, linguagens, ferramentas que visam otimizar e auxiliar a produtividade do desenvolvedor de software. Entretanto, os materiais, geralmente, são espalhados e focados em tecnologias ou linguagens, com pouca explicação conceitual do como ou porquê.

Dessa forma, torna-se necessário uma abordagem mais holística, aliando materiais didáticos com a prática.

Portanto, Guia para o Dev Deslocado é um *ebook* (Livro Digital) *online*, gratuito, desenvolvido com o intuito de demonstrar de forma pragmática práticas de mercado. Aliado a ideia de conseguir guiar o leitor através de explicações conceituais/teóricas, como, também, a partir do desenvolvimento de um aplicativo de mundo real de "exemplo", para que o mesmo saiba ao menos por onde começar a sua aprendizagem dentro do contexto de software.

**Palavras-chaves**: engenharia de software. ebook. projeto de mundo real.

# Lista de Figuras

# Lista de Tabelas

# Sumário

# 1 Introdução

> "O mundo do *software* está sempre mudando. Existem novas tendências surgindo todos os dias. Tendências antigas que agora têm abordagens melhores. Novos problemas a serem resolvidos. Problemas antigos, nunca resolvidos, que agora contam com a tecnologia certa para serem respondidos. Tecnologia antiga que agora traz as respostas. Novos chavões. Palavras-chave antigas. Linguagens, paradigmas, estilos, frameworks, arquiteturas, padrões, metodologias, princípios..."(Carneiro 2020)

A indústria de *software* ainda está em um processo de amadurecimento e constante evolução. Por conta disso, vemos uma grande quantidade de tecnologias, linguagens e ferramentas que visam otimizar e auxiliar a produtividade do desenvolvedor de *software*. Logo, o profissional de *software* precisa estar continuamente se capacitando.

O problema se encontra nos materiais didáticos sobre o assunto, que embora estejam focados em documentação de linguagens e tecnologias, poucos são aqueles que unificam todas as etapas de desenvolvimento. A maioria destes materiais se concentram nas nuânces a respeito de apenas uma linguagem. Fazendo com que, a cada passo dentro do desenvolvimento, seja necessária uma "nova busca" por outro documento que auxilie. Dessa forma, há uma escassez de materiais didáticos (fora do contexto de cursos online) que de fato executem um projeto desde o seu início.

Além desse ponto, há a questão de que, embora o curso de Engenharia de *Software* nos prepare e/ou introduza a diversas áreas de atuação ( *Web*, *DevOps*, Arquitetura e Desenho de *Software*, etc), poucas vezes é possível uma abordagem multidisciplinar que toque em várias dessas áreas.

E, quando isso ocorre, mais raro ainda é a eventualidade dessa prática seguir adiante de forma que tome um corpo realmente rebuscado de "projeto grande" - que aborda questões de *deploy* (mover um *software* para o ambiente de produção), escalabilidade, manutenção & evolução de funcionalidades (funcionalidades ou recursos dentro de um *software*), testes, dentre outras coisas.

Com isso, o objetivo deste trabalho é trazer conceitos atuais adotados no mercado para a construção de *software*, complementando com conceitos apresentados nos currículos acadêmicos. Dessa forma, muito embora cada projeto tenha suas particularidades, a intenção é que todos os elementos dessa jornada sejam percorridos, explicados e "ilustrados", por meio da prática.

## 1.1 Justificativa

### 1.1.1 Problema: Material Multidisciplinar

Doravante, muito embora exista uma infinidade de materiais (práticos e teóricos) na Engenharia de Software, falta-se um material multidisciplinar, pragmático, que aborde os principais conceitos de Engenharia de Software através do desenvolvimento de um único aplicativo.

## 1.2 Objetivos

### 1.2.1 Objetivo Geral

O objetivo principal deste trabalho é construir um *ebook* que possa apoiar profissionais e estudantes de tecnologia, que precisam de um auxílio no desenvolvimento ou que não sabem por onde começar a aprender uma área de Software, já que o processo de desenvolvimento costuma não ser tão simples, principalmente para um programador iniciante, e, também, não é tão claro e nem tão objetivo quanto deveria (Oram e Wilson 2011).

Além disso, a ideia também consiste em trazer a possibilidade de que o trabalho possa ser atualizado pela comunidade; ou seja, tornar o *Ebook* um projeto *Open Source* (Desenvolvimento Aberto, onde o código é livre e pode ser visualizado por qualquer um) para que o conteúdo do livro não fique defasado com o passar do tempo.

Juntamente ao *ebook*, desenvolver uma aplicação de exemplo para que possa ser usada como referência dentro do *ebook*. Dessa forma, os leitores terão uma aplicação prática para reproduzir comandos e conceitos ensinados no livro.

### 1.2.2 Objetivos Específicos

Destacamos os seguintes objetivos específicos que satisfazem a conclusão dos temas apresentados na Seção 1.2.1:

- Realizar estudos em vários âmbitos e áreas que envolvem a engenharia de *Software*;
- Explorar exemplos práticos para sustentar os tutoriais dentro do guia;
- Organizar uma sequência dos conceitos de engenharia de *Software* que deve-se aprender para conseguir construir um produto completo.
- Disponibilizar o guia para o público;
- Formalizar e analisar os resultados obtidos, tanto na aceitação da comunidade (público usuário) quanto no questionário (Seção 3.2.1).

## 1.3 Organização do trabalho

Este trabalho é organizado em 6 capítulos.

- Revisão Bibliográfica (Capítulo 2): falamos sobre alguns motivadores contidos nas literaturas, que nortearam algumas das nossas decisões.
- Metodologia (Capítulo 3): apresenta a forma como levantamos o conhecimento, estruturamos a ideia e o desenvolvimento do projeto.
- Guia para o Desenvolvedor Deslocado (Capítulo 4): explicamos como cada capítulo do Guia foi desenvolvido, evidenciando como cada pesquisa foi feita e como os tópicos foram desenvolvidos.
- Aplicativo (Capítulo 5): falamos acerca da aplicação que foi desenvolvida, com o intuito de dar exemplos práticos de desenvolvimento dentro do guia, além das tecnologias que foram utilizadas.
- Resultados (Capítulo 6): apresentamos dados e informações relevantes acerca da pesquisa que fizemos com ex-alunos de engenharia de *software*, o estado da arte do aplicativo que utilizamos como exemplo, como também a estrutura atual do *ebook*.
- Conclusão (Capítulo 7): Discutimos sobre o trabalho que foi desenvolvido, as escolhas que foram tomadas e, também, perspectivas sobre o futuro.

# 2 Revisão Bibliográfica

## 2.1 Capacitação

Podemos imaginar que, sem a busca pelo aprimoramento e pela capacitação, teríamos profissionais que ficariam estagnados; ou seja, neste cenário, o mercado estaria saturado de pessoas que sabem fazer apenas uma coisa, e apenas na área em que atuam.

> "pesquisadores e praticantes têm percebido que desenvolvimento de *software* (...) é um esforço coletivo, complexo e criativo. Deste modo a qualidade do produto de software depende fortemente das pessoas, organizações e procedimentos utilizados para criá-los e disponibilizá-los" (Fuggetta 2000)

Para FUGGETA, a criação e desenvolvimento de *software* depende do esforço coletivo. E a qualidade do que foi desenvolvido é inerente à qualidade do trabalho/conhecimento dos profissionais que executaram aquelas tarefas. Tendo isso em consideração, ele reforça a necessidade de se investir no material humano, sempre procurando tornar os profissionais aptos a melhor fazer seu trabalho. Seja melhorando procedimentos ou investindo no aprendizado de novas tecnologias.

Sendo a capacitação uma necessidade para aprimoramento pessoal, com a ajuda de algumas fontes de pesquisa e da utilização de ferramentas, o engenheiro passa a se tornar mais eficiente. Mas, onde procurar?

## 2.2 Material Didático

Em nossa busca de materiais para embasar nossos conteúdos no *Ebook*, detectamos 2 principais fontes de materiais: Literatura Acadêmica (2.2.1) e Literatura Cinzenta (2.2.2).

Houve a procura de um conjunto de fontes distintas em livros, artigos, cursos e publicações na internet que dessem apoio ao desenvolvimento de um projeto. Principalmente para se ter uma base tanto conceitual para a produção do *ebook*, como, também, uma base para desenvolvimento da aplicação. Dito isso, cada fonte exercera um papel próprio:

- A literatura acadêmica serviu para embasar conceitos mais técnicos e conceituais do *ebook*.
- A literatura cinzenta, em seus diversos meios, auxiliava-nos a identificar quais as práticas mais atuais no desenvolvimento de aplicações feitas nas *stacks* (conjunto de sistemas necessários para rodar uma aplicação) de tecnologia que optamos por utilizar.

Com isso, procuramos extrair o melhor dos dois mundos, sem ter algo que limitasse a elaboração do guia.

### 2.2.1 Literatura Acadêmica

Rica em informação e conceitos, esse tipo de literatura estimula o aprofundamento teórico do leitor sobre o assunto abordado naquele conteúdo. Ainda que bastante informativa, essa literatura se limita na transmissão de conteúdo para o contexto acadêmico.

> "O engenheiro reúne dispositivos em estruturas capazes de satisfazer uma necessidade humana. A criação de estruturas é essencial para que se extraia uma função útil do conjunto de dispositivos. O desafio do engenheiro de software é escolher e montar as estruturas de grande complexidade que a programação dos computadores permite realizar."(Filho 2000).

Doravante, a necessidade de se atualizar constantemente é muito presente; pois, muito embora conceitos teóricos sejam muito importantes (inclusive facilitando na aprendizagem de novas tecnologias), é necessário buscar por novas soluções ou ferramentas para superar os desafios.

### 2.2.2 Literatura Cinzenta

A partir do que fora entendido com base na literatura acadêmica, passamos então a optar por uma literatura não convencional, conhecida como Literatura Cinzenta.

> "Diz respeito a publicações não convencionais e não comerciais, semi publicadas, difíceis de encontrar em canais tradicionais de distribuição, com controle bibliográfico ineficaz (...), sendo frequentemente não incluídas em bibliografias e catálogos. (...). Apresentam informação e conhecimento altamente atualizados e mais detalhados, alcançam um público reduzido e não são determinadas apenas por interesses comerciais" (Botelho e Oliveira 2017).

Considerando-se que o conhecimento no setor de tecnologia é extremamente compartilhado, aberto e transparente (ainda mais quando considera-se a enorme quantidade de fóruns de discussão e ajuda que são utilizados diariamente por desenvolvedores [e.g. stackoverflow]), basear-se nesse tipo de literatura para o desenvolvimento e aprendizagem de novas tecnologias tende a ser o mais adequado.

Nesse caso, a estratégia que a grande maioria adota para manter-se atualizado no mercado de trabalho é acompanhar as mudanças aprendendo o mínimo de sintaxe e estrutura para poder se virar utilizando a documentação de alguma ferramenta. E, conforme problemas surgem, a comunidade *online* ajuda a resolvê-los (Hora 2021).

É justamente com essa "metodologia orgânica" em mente que este *ebook* foi concebido; isto é, a maior parte de nosso referencial teórico parte de fontes "cinzentas" bem-cotadas (Seção 2.2.2.2).

#### 2.2.2.1 Cursos online de Engenharia de *Software* e materiais parecidos

Encontramos poucos livros que assemelhavam-se com o propósito deste trabalho (um material pragmático que concilia conceitos acadêmicos com práticas de mercado), embora fossem elaborados de outras formas. Encontramos, também, alguns cursos que eram voltados a projetos, cursos esses que eram análogos à nossa proposta, — porém, todos eram pagos.

Outros materiais orientavam de uma forma geral o que fazer em um projeto de engenharia de *software* e/ou o papel de um engenheiro de *software*. Entretanto, mesmo sendo gratuitos e de boa qualidade, também não eram, necessariamente, orientados a um projeto único que amarrasse todos os conceitos de forma pragmática.

#### 2.2.2.2 Critérios de aceitação para fontes

Seguindo insights advindos do artigo "Strategies For Grey Literature Review" (Wen et al. 2020), estabelecemos que, no geral, utilizaríamos fontes que atendessem a, ao menos, dois dos requisitos abaixo:

- Ser recente (2018 em diante)
- Ter boa avaliação da comunidade (*"claps"* no Medium, *"likes"* no Youtube, além de comentários no geral)
- Ser respeitada (pessoas de renome, como Martin Fowler, ou organizações renomadas, como Mozilla)

Com esses critérios em mãos, passou-se a explorar novas fontes de conteúdo com uma maior cautela, para proporcionar qualidade necessária para ser incrementado ao *Ebook*.

#### 2.2.2.3 Materiais utilizados

Pode-se visualizar, abaixo, alguns dos materiais *cinzentos* que foram utilizados na construção deste trabalho.

Eis alguns dos cursos que encontramos:

Tabela 1 – Cursos de Software

| Curso | Link |
|---|---|
| OneBitCode | <https://onebitcode.com/cursos/> |
| Learn Software Engineering with Online Courses and Lessons | <https://www.edx.org/learn/software-engineering> |
| Curso Desenvolvimento Web Full Stack | <https://www.digitalhouse.com/br/curso/desenvolvimento-web-full-stack> |
| Desenvolvimento Web Completo 2021 - 20 cursos + 20 projetos | <https://www.udemy.com/course/web-completo/> |
| Dev Full Stack | <https://hsmuniversity.com.br/cursos/code/dev-full-stack/> |
| Become a Full-Stack Software Engineer - Kick-Start Your Career | <https://www.codecademy.com/learn/paths/full-stack-engineer-career-path?g> |
| Desenvolvedor Full-Stack - Impacta | <https://www.impacta.com.br/carreira-full-stack/> |
| Bootcamp de Programação Web - Responde Aí | <https://page.respondeai.com.br/bootcamp> |
| Escola online para desenvolvedores | <https://www.treinaweb.com.br> |
| Alura - cursos oline de tecnologia | <https://www.alura.com.br/> |

Outros materiais, cujo quais orientavam de uma forma geral o que fazer em um projeto de engenharia de *software* e ou o papel de um engenheiro de *software*:

Tabela 2 – Posts de Software

| Post | Link |
|---|---|
| My Software Engineer Roadmap | <https://medium.com/swlh/my-software-engineer-roadmap-2fb0c02b8a08> |

| Post | Link |
|---|---|
| SWEBOK - Guide to the software engineering body of knowledge | <https://www.computer.org/education/bodies-of-knowledge/software-engineering> |
| Software Engineer Roadmap | <https://github.com/kamranahmedse/developer-roadmap> |
| freeCodeCamp.org | <https://www.freecodecamp.org/> |

## 2.3 Construção dos Capítulos

À parte da capacitação conceitual, ponderamos como iríamos estruturar os capítulos do *ebook*; para tanto, utilizamo-nos de algumas ideias de um filósofo alheio à área de Engenharia de Software, mas que muito têm a agregar.

### 2.3.1 Mortimer Adler

Mortimer Jerome Adler foi um filósofo, educador, enciclopedista e autor de livros. Suas maiores obras foram "Como ler livros", "Aristóteles para todos" e "Como falar, como ouvir".

Os trechos a seguir advém do primeiro livro:

> "Porém, gostaríamos de alertar para o fato de que muitas pessoas - até mesmo leitores experientes - em geral desprezam o valor da leitura inspecional. Elas começam a leitura na primeira página e avançam diligentemente até a última, sem ao menos ler o sumário. Assim, elas se defrontam com o desafio de extrair um conhecimento superficial do livro ao mesmo tempo que tentam entendê-lo. É uma dificuldade e tanto." (Adler e Van Doren 1972)

> "No intuito de defender seu argumento, Aristóteles mostra como a unidade da Odisseia pode ser resumida em poucas frases: 'Um homem solitário vagueia, durante anos, em terras estrangeiras; ele é vigiado pelo ciumento Possêidon, que o impede de voltar, e fica desolado. Em casa, os pretendentes de sua esposa lhe devoram os bens e ameaçam a vida de seu filho. Quando, finalmente, consegue regressar, ele revela a alguns sua identidade, ataca e destrói os inimigos com as próprias mãos e salva-se.' 'Eis a parte essencial', diz Aristóteles, 'tudo o mais são episódios.' " (Adler e Van Doren 1972)

Pautado nas ideias de Mortimer Adler (e Aristóteles), inferimos que:

- A maior demonstração de compreensão acerca de um assunto é a capacidade de demonstrar o mesmo a partir de uma unidade única de uma ideia de forma sintetizada, ou seja da forma mais simples e objetiva possível.
- Os leitores deveriam se preparar para ler o conteúdo antes de começar a consumir o livro, verificando os tópicos contidos no sumário, para que haja um preparo e não receba um "choque" ao se confrontar com o conteúdo.

### 2.3.2 Resultado

A estrutura final da construção dos capítulos pode ser observada na Seção 6.2.1.

# 3 Metodologia

Esse capítulo tem o intuito de esclarecer a forma que nos organizamos para o desenvolvimento das três frentes desse trabalho: TCC, *Ebook* e Aplicativo.

## 3.1 TCC

### 3.1.1 Organização do Trabalho: Github

Para a alocação de tarefas, discussão de problemas e melhorias, e principalmente como repositório remoto Git, utilizamos o serviço Github. Em nossa organização (github.com/appointment-octopus), centralizam-se todas as frentes de trabalho que abordamos.

## 3.2 Desenvolvimento do *Ebook*

### 3.2.1 Pesquisa com Ex-Alunos

Conduzimos uma pesquisa com alunos formados em Engenharia de Software pela UnB. Essa pesquisa, tinha como principal objetivo saber o quão bem a universidade preparou cada um deles para o mercado de trabalho. Nessa pesquisa procuramos saber algumas informações a respeito das experiências pós-universidade que cada um desses alunos teve.

Então, a pesquisa era composta por perguntas que envolviam:

- o que os alunos estão utilizando atualmente no mercado de trabalho
- o que sentiram falta na graduação
- conteúdos que julgavam ter sido de grande valia durante a graduação.

Para tanto, enviamos um email para cada um dos ex-alunos solicitando a participação através de um formulário. Vale ressaltar que as respostas coletadas foram anônimas, para que o participante se sentisse confortável a fazer uma crítica transparente sobre o tempo em que esteve na universidade.

### 3.2.2 Desenvolvimento dos capítulos

No começo do desenvolvimento, foram levantados os primeiros assuntos que falaríamos a respeito, logo em seguida, dividimos estes entre os membros para que cada um fosse responsável por gerar conteúdo acerca daquele tema. Em seguida, o membro que não era responsável pela criação daquele material torna-se responsável pela revisão, leitura e validação do material; dessa forma, o conhecimento sobre aquele tema é difundido entre os membros, e,

também, é possível obter um "olhar externo" (de quem não esteve envolvido na construção daqueles conteúdos), para se ter uma ideia se seria bem aceito pelo público.

### 3.2.3 Implementação do livro

Após as questões mais teóricas, surgiu a questão: onde escrever esse livro?

Já que gostaríamos que ele tivesse a nossa cara e a nossa dinâmica, tentamos várias abordagens.

A primeira, era de fazer o livro utilizando o *framework* (Estrutura de códigos que facilitam o desenvolvimento) de *frontend* (Interface gráfica de um projeto) Svelte para poder produzir as interfaces do livro, para que o usuário pudesse navegar pelas páginas de forma fluida. Entretanto, essa tarefa acabou se demonstrando um pouco mais árdua e complexa do que fora antecipado, já que muita coisa teria de ser feita (e refeita), e acabaríamos reinventando a roda.

Depois de superarmos essa etapa, pesquisamos pelo Gitbook e vimos como uma possibilidade, entretanto não era exatamente gratuito e, portanto, fora descartado.

Por fim, encontramos o mdBook, que é feito em Rust (e inclusive foi utilizado para fazer a documentação da linguagem Rust). Optamos por utilizar essa ferramenta principalmente pela facilidade de transformar *Markdown* para uma linguagem de interface web, além de já renderizar um livro elegante com vários recursos customizáveis (como tema claro/escuro, dentre inúmeras outras coisas).

## 3.3 *App*

O *Appointment octopus* foi o nome dado a aplicação que desenvolvemos para ilustrar alguns capítulos do *ebook*, para que o leitor do *ebook* tenha uma experiência mais ativa.

Dividimos o projeto em serviços e separamos as funcionalidades para serem desenvolvidas por cada membro. Dessa forma, optamos por desenvolver primeiro para, depois, explicar e introduzir no *ebook* conforme fosse necessário.

# 4 Guia do Desenvolvedor Deslocado

Pode-se acessar *ebook* atualmente pelo link: https://appointment-octopus.github.io/ebook/

O principal objetivo é que o livro seja um material didático complementar de apoio. Também, que aborde os principais tópicos da engenharia de software de uma forma mais pragmática para que iniciantes no mundo de software saibam o que e quando aprender, embasando as escolhas técnicas e explicando, quando possível, outras abordagens viáveis também. Para o desenvolvimento do *ebook* seguimos os seguintes passos:

1. Assuntos Chaves (4.1): Mapear palavras/assuntos/áreas chaves que deveríamos considerar para o desenvolvimento do livro.
2. Primeira Estrutura do *Ebook* (4.2): Estruturar estes conteúdos em uma linha do tempo simples, como um passo-a-passo.
3. Capítulos do *Ebook* (4.3): Definir os capítulos, e como estes seriam desenvolvidos num formato mais dinâmico (e menos conteudista), de forma a ser mais engajador.

## 4.1 Assuntos Chaves

Primeiramente, decidimos pesquisar as grandes áreas de engenharia de software, tendências, ferramentas etc. Destacam-se as seguintes fontes:

Tabela 3 – Fontes para Assuntos-Chaves

| Título | Fonte |
|---|---|
| "Software Engineer Roadmap" | <https://github.com/kamranahmedse/developer-roadmap> |
| "Technology Radar - An opinionated guide to technology frontiers - ThoughtWorks" | <https://www.thoughtworks.com/radar> |
| "A Survey of DevOps Concepts and Challenges" | DOI: 10.1145/3359981 |
| "Guide to the software engineering body of knowledge" | ISBN: 9780769551661 |
| "How To Build a Large Software Project Alone, From Scratch" | <https://betterprogramming.pub/how-to-build-alone-a-large-software-project-from-scra |
| "My Software Engineer Roadmap" | <https://medium.com/swlh/my-software-engineer-roadmap-2fb0c02b8a08> |

De antemão, vale ressaltar que vários conceitos/palavras-chaves eram estranhos a nós mesmos, sendo estes elencados aqui, também, como um guia de estudo que deveríamos aprofundar-nos mais. A partir disso, estruturamos as tabelas abaixo elencando os conceitos/ferramentas que eram citados com maior frequência (nas respectivas grandes áreas).

### 4.1.1 *Devops*

Necessário conhecer Redes, Linux, Bancos de Dados, Integração Contínua/*Deploy* Contínuo (CI/CD), *Logging* e Monitoramento dos serviços, Nuvem... entre outras coisas. Os principais conceitos/ferramentas foram elencados abaixo:

Tabela 4 – *DevOps*

| Palavra-chave | Contexto |
|---|---|
| 1. linux, comandos, bash, ssh | server adminstration |
| 2. tcp/ip, cdn, dns, http2 (tcp), http3 (udp), https, ftp, ssl, firewall | network & security |
| 3.1. apache tomcat/nginx, reverse proxy, proxy (circuit breaker [hystrix]), load balancer | web server |
| 3.2. redis/hazelcast e varnish (frontend) | caching |
| 3.3. mongo/cassandra, postgres/mysql, NewSQL (google spanner, cockroachdb) | databases |
| 3.4. apache kafka/rabbitmq | message broker |
| 3.5. apache spark/hadoop | parallel data processor |
| 3.6. apache flink/aws lambda | data streaming |
| 3.7. snowflake | data platform |
| 4.1. ansible | configuration management |
| 4.2. docker, CRI, OCI | containerization |
| 4.3. kubernetes (e helm?, openshift?) | container orchestrator |
| 4.4. terraform | infrastructure as code |
| 4.5. gitops (weaveworks) | guideline |
| 5. jenkins/circle ci/travis/gh actions | ci/cd |

| Palavra-chave | Contexto |
|---|---|
| 6.1. prometheus/zabbix/nagios | monitoring |
| 6.2. ELK (elasticsearch, logstash, kibana) | logging |
| 6.3. zipkin/jaegger | tracing latency |
| 6.4. gatling test | observability |
| | |
| 7.1. aws e serverless | cloud |
| 7.2. minio | multicloud |
| 7.3. pivotal, anthos, rancher | cloud-agnostic frameworks |

### 4.1.2 Conceitos gerais

Boas práticas na engenharia de software.

Tabela 5 – Assuntos Gerais

| Palavra-chave | Contexto |
|---|---|
| SOLID | Principios de Design de Software |
| YAGNI | Jargão |
| Clean Code | Boas práticas para escrita e compreensão do código |
| sonarqube, jfrog, code climate, codacy | Ferramentas de análise estatica da qualidade do código |

### 4.1.3 *Design* e *Frontend*

Como fazer uma interface com boa UI/UX (interface de usuário, experiência de usuário)? Ou seja, algo com boa aparência estética e boa interatividade?

Tabela 6 – Design

| Palavra-chave | Contexto |
|---|---|
| neuromorphism | trend, novo estilo de UI |
| motion design | animação para portifólio, “trailer” do protótipo |
| a11y | acessibilidade |

| Palavra-chave | Contexto |
|---|---|
| figma | ferramenta pra prototipar |

Tabela 7 – *Frontend*

| Palavra-chave | Contexto |
|---|---|
| storybook | ferramenta para desenvolver componentes “isoladamente” |
| microfrontends, mobile-first, AMP, RWD, PWA | design patterns para frontend |
| RAIL model | modelo de performance |
| next.js, svelte, react, vue, angular | ferramentas para desenvolver |

### 4.1.4 *Backend*

Principais ferramentas e estruturas para criação de uma API.

Tabela 8 – *Backend*

| Palavra-chave | Contexto |
|---|---|
| REST, SOAP, ATOM, CQRS, gRPC (protobuf), graphQL, webassembly | formas de estruturar a api |
| deno/nest.js, spring, gorilla, .net, elixir, clojure | ferramentas para desenvolver |
| swagger, AsyncAPI | documentação da api |
| BFF (backends for frontends) | separação das apis |

### 4.1.5 Segurança

Conceitos necessários para assegurar os dados da aplicação

Tabela 9 – Segurança

| Palavra-chave | Contexto |
|---|---|
| SAST e DAST | teste de segurança da aplicação (estatico e dinâmico) |
| SQL Injection, DDOS attack | técnicas de ataque comum |
| sha256 + bcrypt + aes256 + pepper | tipos de encriptação |
| backup | técnicas para resguardar dados e aplicações |

## 4.2 Primeira Estrutura do *Ebook*

Então, tendo as tabelas anteriores em mente, concebemos um fluxograma que introduziria cada conceito de forma iterativa, passo-a-passo, passando de um conceito pro próximo.

O mesmo se encontra descrito abaixo; entretanto, para melhor visualizá-lo clique aqui.

---

1. **Introdução**

- 1.1. Propósito do livro
- 1.2. Processo/ciclo de desenvolvimento proposto, de forma simplificada
- 1.3. Diagrama do *flow* do livro

2. **Glossario**

3. **Dicas gerais**

- 3.1. Como googlar
  - 3.1.1. String de Busca: usar palavras-chaves
  - 3.1.2. Operadores: Aspas, Menos, Ou, Range, Site, Defina, Asterisco, Filetype, Dois pontos
- 3.2. O que é git
  - 3.2.1. Utilidade
  - 3.2.2. O que é github
  - 3.2.3. Comandos básicos
  - 3.2.4. Comandos subestimados/subutilizados
  - 3.2.5. *Flow* de desenvolvimento, *deploy* etc
- 3.3. Linux
  - 3.3.1. Estrutura do sistema
  - 3.3.2. Bash/shell
    - 3.3.2.1 Comandos Básicos: cp, cat, diff, mv, rm, echo, chmod, find, sed
    - 3.3.2.2 Estrutura Bash: .bashrc, .bash_profile, .bash_history
  - 3.4. Network

        * 3.4.1. CDN
        * 3.4.2. DNS
        * 3.4.3. TCP e UDP
        * 3.4.4. HTTP2, HTTP3, HTTPS
        * 3.4.5. FTP
        * 3.4.6. SSL
        * 3.4.7. Firewall
        * 3.4.8. Ssh
    - 3.5. Vscode
    - 3.6. S.O.L.I.D, clean code
    - 3.7. Regex
    - 3.8. Extras
        * 3.8.1. Estrutura de dados, algoritmos, complexidade
        * 3.8.2. Contratação
            · 3.8.2.1. Curriculo
            · 3.8.2.2. Linkedin
            · 3.8.2.3. Processo Seletivo
        * 3.8.3. Unicode, UTF-8

4. **Documentação**

- 4.1. Elicitação de requisitos
    - 4.1.1. Briefing & Entrevista
    - 4.1.2. Questionário
    - 4.1.3. Doc de visão & ZeBrand
    - 4.1.4. Casos de uso
- 4.2. Prototipo/Design
    - 4.2.1. Inspiração em outras fontes
    - 4.2.2. Espaço negativo [respiro]
    - 4.2.3. Grid e Alinhamento
    - 4.2.4. Proporção 6:3:1
        * 4.2.4.1. Cores primárias
        * 4.2.4.2. Cores de acentuação
        * 4.2.4.3. Escala de cinzas
            · 4.2.4.3.1. Modelo HSB, HEX, RGB etc

  - 4.2.5. Usabilidade
    - 4.2.5.1. Navegabilidade; Affordances e Signifiers (sugestividade)
    - 4.2.5.2. Familiaridade/Comunicabilidade
    - 4.2.5.3. Tolerancia de erros
  - 4.2.6. Arquitetura da informação
  - 4.2.7. Prototipo de papel
  - 4.2.8. Prototipo de alta fidelidade
    - 4.2.8.1. Figma
      - 4.2.8.1.1. Dicas figma
    - 4.2.9. Wireframe
    - 4.2.10. Tipografia
      - 4.2.10.1. Escala
      - 4.2.10.2. Legibilidade
      - 4.2.10.3. Escolha de fontes
  - 4.2.11. Teste de Usabilidade
- 4.3. Estruturação da Implementação
  - 4.3.1. Doc de arquitetura
  - 4.3.2. Diagrama de pacotes
  - 4.3.3. Diagrama de classes
  - 4.3.4. Bancos
    - 4.3.4.1. Mer
    - 4.3.4.2. Der
- 4.4. Priorização/MoScoW
- 4.5. Processos
  - 4.5.1. Kanban
  - 4.5.2. Bpmn

5. **Implementação dos microsserviços**

- 5.1. Estrutura do desenvolvimento
- 5.2. Básico *devops*
  - 5.2.1. Configuração ambiente
  - 5.2.2. Configuração guia de estilo

  - 5.2.3. Configuração qualidade de codigo
    - 5.2.3.1. Como coletar cobertura de testes
    - 5.2.3.2. Ferramentas disponíveis
      - 5.2.3.2.1. Codecov, Code Climate, Codacy, Embold
      - 5.2.3.2.2. Sonarqube, JFrog, Analizo
  - 5.2.4. Configuração gh actions, circleci, travis etc

- 5.3. *Back*
  - 5.3.1. Documentação/uso das Apis: Swagger/AsyncAPI [citar insomnia postman?]
  - 5.3.2. *Back* auth
    - 5.3.2.1. Controlar requests dos usuários: Redis
  - 5.3.3. *Back* pagamento
  - 5.3.4. *Back* consultas
  - 5.3.5. Gateway: Graphql

- 5.4. *Front*
  - 5.4. Css
    - 5.4.1. CSS Grid
    - 5.4.2. CSS Flexbox
    - 5.4.3. Sass
    - 5.4.4. Bootstrap, bulma, houdini, tailwind...
  - 5.5. Biblioteca de componentes
    - 5.5.1. Styled components
    - 5.5.2. Atomic design
    - 5.5.3. Storybook
    - 5.5.4. Testes: jest & enzyme
  - 5.6. Mobile
    - 5.6.1. Aplicativo usuario
    - 5.6.2. Aplicativo admin
  - 5.7. Web
    - 5.7.1. Web usuario
    - 5.7.2. Web admin
  - 5.8. Validação
    - 5.8.1. Checklist a11y para verificar acessibilidade

        * 5.8.2. Velocidade: Google Lighthouse

6. ***Devops* Produto**

- 6.1. Heroku, netflify, now.sh etc
- 6.2. Web Server
  - 6.2.1. Nginx
    * 6.2.1.1. Proxy reverso
    * 6.2.1.2. Load Balancers
- 6.3. Aws, Digital ocean
- 6.4. Message Broker: Rabbitmq
- 6.5. Configuration Management: Ansible
- 6.6. Infrastructure as Code: Terraform
- 6.7. Monitoramento: Prometheus
- 6.8. Logging: ELK
- 6.9. Orquestrador: Kubernetes

---

## 4.3 Capítulos do *Ebook*

Entretanto, enquanto escrevíamos, percebemos que essa estrutura fazia com que o *ebook* se tornasse maçante, já que possuía muito conteúdo teórico e quase nenhuma prática; e, considerando-se que um dos maiores objetivos do *ebook* era o de engajar e animar os leitores com a ideia de incrementalmente construir coisas palpáveis, decidimos alterar a estrutura previamente proposta.

Embora alguns capítulos ainda não tenham sido concluídos, deve-se ter-se em mente que o livro está em constante "crescimento", sofrendo adições e remoções. No futuro, esperamos, também, que a comunidade desempenhe um papel nessa manutenção constante dos conteúdos.

Eis o fluxograma dos capítulos atuais:

---

1. **Introdução**

- 1.1. Sofisticação Técnica - Como Se Virar

### 4.3.1 Sofisticação Técnica - Como Se Virar

Esse capítulo nasceu com a intenção de ensinar algumas formas de aprender sozinho.

Embora só tenha um subcapítulo, que envolve "saber googlar" (*Como Conseguir Melhores Resultados de Pesquisa | Seja Um Profissional Em Pesquisas No Google*, [s.d.]), tem como objetivo trazer, também, um conjunto variado de possibilidades para otimizar buscas e pesquisas *online* em ambientes mais acadêmicos, por exemplo.

Isso é para que o leitor consiga exercitar a capacidade de autonomia para desvendar as formas de sanar o seu próprio problema, antes que seja necessário buscar pelo apoio de terceiros.

### 4.3.2 Linux

Por se tratar de um Sistema Operacional que representa uma certa minoria no mercado (quando incluímos quem não é desenvolvedor), é importante dar uma atenção maior a suas qualidades (Chakon 2017) e, por ser mais utilizado para desenvolvimento, optamos por trazer um conjunto de conhecimentos acerca dessa ferramenta, já que esta poderia vir a ser o ambiente principal de trabalho, além de algumas informações sobre a variedade de distribuições utilizada pela comunidade.
Tendo isso em mente, procuramos nos aprofundar nas seguintes questões acerca deste tópico:

#### 4.3.2.1 O que é

Considerando que queremos explicar o porquê da utilidade da ferramenta e convencer o leitor de que é uma boa opção, este subcapítulo se trata de uma explicação não tão técnica, abordando, então, questões como o porquê de se utilizar, como nasceu e como a comunidade sustenta e evolui o Linux com o passar do tempo. Logo, nesse subtópico explicamos desde o surgimento até uma breve descrição a respeito das distribuições de Linux.

#### 4.3.2.2 Windows WSL

Este subtópico é mais uma sugestão para quem não quer ter de trocar de ambiente, ou, que não pretendem recorrer a um *dual boot*. Em suma, é baseado no próprio tutorial da Microsoft, ensinando como instalar e manusear esta ferramenta, que se comporta como um terminal do Linux (mas no Windows).

#### 4.3.2.3 *Dual Boot*

Considerando os grandes receios e dúvidas de quem nunca se desbravou em tentar realizar um *dual boot* em sua máquina, — ou, pior, alguém que nunca utilizou outro sistema operacional —, decidimos criar um capítulo reservado a isso, com o intuito de desmistificar essa questão e mostrar que, talvez, isso seja menos pavoroso do que parece.

#### 4.3.2.4 Terminal

É provável que seja o principal subtópico deste capítulo, sendo responsável por potencializar ao máximo a utilização de um Linux. Aqui será explicado como funciona o terminal, e conterá a maior parte dos principais comandos possíveis, explicando como funcionam e formas de aplicabilidade. Desta forma, procuramos incentivar o leitor a reproduzir os comandos em seu próprio computador. No momento, é discutido, por exemplo, como navegar entre diretórios, manipulação de arquivos através do terminal, alguns atalhos úteis para melhor manuseio, etc.

### 4.3.3 Git

A principal ferramenta para qualquer desenvolvedor moderno, tendo em vista que é inconcebível o desenvolvimento de algo em larga escala sem o versionamento e compartilhamento de código. Este capítulo foi desenvolvido tendo-se como base os seguintes materiais:

Tabela 10 – Materiais sobre Git

| Material | Link |
|---|---|
| Practical Git: confident Git through practice | amazon.com/Practical-Git-Confident-Through-Practice/dp/1484262697 |
| Learn Enough Git to Be Dangerous | <https://www.softcover.io/books/28fdb94f/learn_enough_git> |
| Advanced git techniques | <https://elib.dlr.de/139030/> |
| New Developer? You should've learned Git yesterday. | <https://codeburst.io/number-one-piece-of-advice-for-new-developers-ddd08abc8bfa> |
| Git: A Complete Guide | <https://towardsdatascience.com/git-a-complete-guide-d49675d02a5d> |
| Developing Open Source Software using Version Control Systems: An Introduction to the Git Language for Documenting Your Computational Research | <https://digitalcommons.usu.edu/ecstatic_all/87> |

| Material | Link |
|---|---|
| How to use Git efficiently | <https://medium.com/free-code-camp/how-to-use-git-efficiently-54320a236369> |
| Learn Git Version Control using Interactive Browser-Based Labs - Katacoda | <https://www.katacoda.com/courses/git> |
| "Resources to learn Git" | <https://try.github.io/> |
| "Version Control Best Practices for Collaboration" | <https://learn.gitlab.com/version-control/version-control-best-practice> |

A divisão dos tópicos fora definida da seguinte forma:

- Git & Github
- Instalação e Configuração
- Principais comandos

#### 4.3.3.1 Git & Github

Fazendo uma diferenciação a respeito de ambos, da importância de cada um e introduzindo ao novo tema.

#### 4.3.3.2 Instalação e Configuração

Uma explicação acerca da instalação e configuração da ferramenta, para customizar de forma mais pessoal.

#### 4.3.3.3 Principais comandos

Um conjunto dos principais comandos (como `git status`, `git add`, `git commit` etc), flags utilizáveis em cada um destes, exemplos práticos de `merge`, `rebase` entre outras coisas.

### 4.3.4 Ambientação

Nesse capítulo buscamos abordar assuntos que envolviam a arquitetura cliente-servidor, Docker, desenvolvimento de projeto e contribuição com o *Open Source*. Assim, para tentar auxiliar na ilustração dessa temática, utilizamos um projeto pronto de "mundo-real" ("Conduit - RealWorld example App" 2021) para dockerização e configuração de um *backend* & *frontend*. Isso para que o usuário tenha um exemplo prático a parte de como poderá fazer essa configuração sem muito mistério.

Eis os assuntos que foram abordados:

#### 4.3.4.1 Arquitetura cliente-servidor

Explicamos o conceito de cliente servidor de forma simples, tendo em vista que era oportuno já que teríamos um serviço para o *frontend* (cliente) e um para o *backend* (servidor). Neste tópico explicamos como isso é feito com exemplos (em alto nível de abstração) de como seria em código, explicitando como essa conversa aconteceria em uma aplicação fictícia.

#### 4.3.4.2 Projeto Javascript

Neste ponto saímos da parte teórica de como se comporta essa situação, e passamos para manipulação e compreensão de um projeto completamente desenvolvido em Javascript. Como instalar as dependências, como funciona a estrutura de um aplicativo Javascript, como rodar a aplicação, dentre outras coisas. Com isso conseguimos abordar tanto o *frontend* quanto o *backend*.

***Backend***

O código do serviço é analisado juntamente com o leitor. Também é ensinado como consumir a *api* com ferramentas como Postman/Insomnia.

***Frontend***

Dá continuidade aos comandos abordados no tópico acima, fazendo com que o *front* consiga se comunicar com os bancos das *api* locais.

#### 4.3.4.3 Docker

Neste capítulo, explicamos a fundo o que é Docker. Desde sua utilização básica, passando pela construção de um container mínimo (em linux) para apreensão de conceitos que possibilitam a existência dessa ferramenta; também distinguimos *VMs* (*Virtual Machines*, ou, Máquinas Virtuais [tradução livre]) e *containers*, além de explicar a fundo (e exemplificar) os comandos que podem ser utilizados no Dockerfile.

### 4.3.5 FAQ

Há de se ressaltar que vários capítulos possuirão um espaço para *FAQ* (Frequently Asked Questions, ou, Perguntas Frequentemente Feitas [tradução livre]), constituído nas perguntas e dúvidas que mais se repetem em fóruns como o *StackOverflow*. Estes capítulos são gerados de forma semi-automática.

# 5 Aplicação

Neste capítulo será apresentado o processo de construção do aplicativo que é utilizado de exemplo no *Ebook*.

Inicialmente, eis a definição do que seria o aplicativo:

> *Um aplicativo pessoal, para um médico especifico (o cliente), de forma que houvesse um controle da fila de espera de seus pacientes. Os pacientes baixariam o aplicativo, pagariam a consulta (somente boleto/credito, sem plano de saúde e afins), sem data marcada, e entrariam na fila. Os pacientes poderiam acompanhar, pelo aplicativo, em qual posição da fila estavam.*

## 5.1 Introspecção - Mapeamento de funcionalidades

Posteriormente, ao longo de reuniões fora decidido mudar o foco inicial, tendo como resultado um novo escopo: um aplicativo para usuário (onde agendará as consultas) e um aplicativo para administrador (onde gerenciará os pacientes, consultas e afins).

Ainda destoando da proposta inicial, a ideia de fila foi removida. Definimos que os horários e dias seriam disponibilizados pelo Admin. Sendo assim, o usuário pode escolher entre os dias/horários disponíveis para agendar sua consulta.

O Admin poderá definir tipos de consultas, cada qual com seu próprio preço e tempo de duração (se necessário) estipulado.

Também foi convencionado que o fluxo do aplicativo do usuário deveria ser simples: `Escolha de data -> "Carrinho" para verificação do pedido -> Pagamento -> Tela de confirmação do agendamento.`

Fora isso, outras ideias foram discutidas mas deixadas de lado por ora; como, por exemplo:

- Existência de um chat para comunicação entre Paciente e Admin
  - Chamadas e/ou Video-chamada
- Usar bibliotecas de OCR (Optical Character Recognition) para preencher dados do cartão (de crédito/débito) do usuário através da câmera fotográfica
- Gerar PDF com nota fiscal

## 5.2 Especificação/Documentação

A documentação que envolvia principalmente o desenvolvimento da aplicação, possui a sua estrutura inicial armazenada no *git*.

Para detalhar e chegar ao caminho que deveria ser trilhado, fizemos um protótipo e casos de uso para validar e entender melhor o contexto e o problema que, embora não fosse real, nós nos propúnhamos a resolver.

### 5.2.1 Protótipo

Como forma de não só levantar mais requisitos, mas, também, de visualizar outras possíveis funcionalidades, optamos pela criação de um protótipo de alta fidelidade, criado no Figma.

A criação do protótipo nos auxiliou, também, em outras questões que envolviam um planejamento estético para que fosse concebido uma identidade visual do produto. Embora não fosse o foco, percebemos, ao longo de reuniões de planejamento, que se fazia necessário ter algo palpável, para que pudéssemos ter uma noção de onde gostaríamos de chegar.

#### 5.2.1.1 Tipografia & Paleta de Cores

As fontes selecionadas foram:

Tabela 11 – Tipografia

| Nome | Fonte | Estilo | Tamanho |
|---|---|---|---|
| Header 1 | Montserrat | Bold | 44 |
| Header 2 | Montserrat | Bold | 34 |
| Header 3 | Montserrat | Bold | 26 |
| Header 4 | Montserrat | Bold | 20 |
| Subtitle 1 | Fira Sans | Medium | 18.5 |
| Subtitle 2 | Fira Sans | Medium | 17 |
| Subtitle 3 | Fira Sans | Medium | 13 |
| Button 1 | Quattrocento Sans | Bold | 22 |
| Button 2 | Quattrocento Sans | Bold | 18.5 |
| Button 3 | Quattrocento Sans | Bold | 14.5 |
| Body 1 | Hind Madurai | Regular | 18.5 |
| Body 2 | Hind Madurai | Regular | 17 |
| Body 3 | Hind Madurai | Regular | 12 |

Já a paleta de cor ficou definida da seguinte forma:

Tabela 12 – Paleta de Cores

| Tipo de cor | Hex |
|---|---|
| Escala de Cinza (Claro 1) | #F2F5FF |
| Escala de Cinza (Claro 2) | #C3CBE5 |
| Escala de Cinza (Médio 1) | #99A2BF |
| Escala de Cinza (Médio 2) | #7480A6 |
| Escala de Cinza (Médio 3) | #54618C |
| Escala de Cinza (Escuro 1) | #333F66 |
| Escala de Cinza (Escuro 2) | #111D42 |
| Background | Gradiente, de “Médio 1” até “Claro 2” |
| Cor Primária 1 | #3764B3 |
| Cor Primária 2 | #8F47AE |
| Cor Primária 3 | #423E94 |
| Cor de Acentuação | #066CE5 |
| Cor de Controle (positivo) | #0E9594 |
| Cor de Controle (negativo) | #780116 |

Pode ser melhor visualizado no Anexo 1 (Seção 8.1).

### 5.2.2 Casos de Uso

Ainda numa fase de contextualização inicial, para que fosse descrito de forma clara, utilizamos da técnica de Casos de Uso, para entender o real fluxo do usuário dentro da aplicação, e conseguir, também, evidenciar funcionalidades que deveriam ser priorizadas, melhoradas ou removidas. Utilizamos a ferramenta PlantUML, que auxiliava na criação e, se necessário, na edição e refatoração de uma imagem que por sua vez gera um diagrama de casos de uso.

A Figura 1 ilustra como se comporta a ferramenta, começando com a declaração das ações e como estas se relacionam. Criamos 2 diagramas de Casos de Uso. O primeiro diagrama (Figura 3) fora criado com o intuito de ilustrar o fluxo da aplicação, destinada ao paciente (usuário), como um todo (desde a primeira vez em que realizou um login). Já o segundo diagrama (Figura 2) ilustra as funcionalidades que serão necessárias para que o usuário possa editar e customizar a sua página de perfil.

```
@startuml profile

skinparam packageStyle rectangle

actor :Logged in\nUser: as user

rectangle "Profile" {

    (UC01 Open profile menu) as open_profile_menu

    (UC02 Edit name) as edit_name
    (UC03 Edit date of birth) as edit_dof
    (UC04 Edit gender) as edit_gender
    (UC05 Edit email) as edit_email
    (UC06 Edit cpf) as edit_cpf

    (UC07 View registered\nbank cards) as view_cards
    (UC08 Open "more" menu) as open_more_menu

    (UC09 Add new photo) as add_new_photo
    (UC10 Remove photo) as remove_photo
    (UC11 Logout) as logout

    (UC12 Add new card) as add_new_card
    (UC13 Remove card) as remove_card
    (UC14 Edit card) as edit_card

    user -up->  open_profile_menu

    open_profile_menu ..> edit_name : << includes >>
    open_profile_menu ..> edit_dof : << includes >>
    open_profile_menu ..> edit_gender :
 << includes >>
    open_profile_menu ..> edit_email :
 << includes >>
    open_profile_menu ..> edit_cpf : << includes >>

    open_profile_menu ..up..> view_cards :
 << includes >>
    open_profile_menu ..up..> open_more_menu :
 << includes >>

    open_more_menu <.up. add_new_photo :
 << extends >>
    open_more_menu <.up. remove_photo :
 << extends >>
    open_more_menu <.up. logout : << extends >>

    view_cards .up.> add_new_card : << includes >>
    view_cards .up.> remove_card : << includes >>
    view_cards .up.> edit_card : << includes >>

}

@enduml
```

Figura 1 – Ferramenta Plantuml - Especificação para geração do diagrama de Perfil

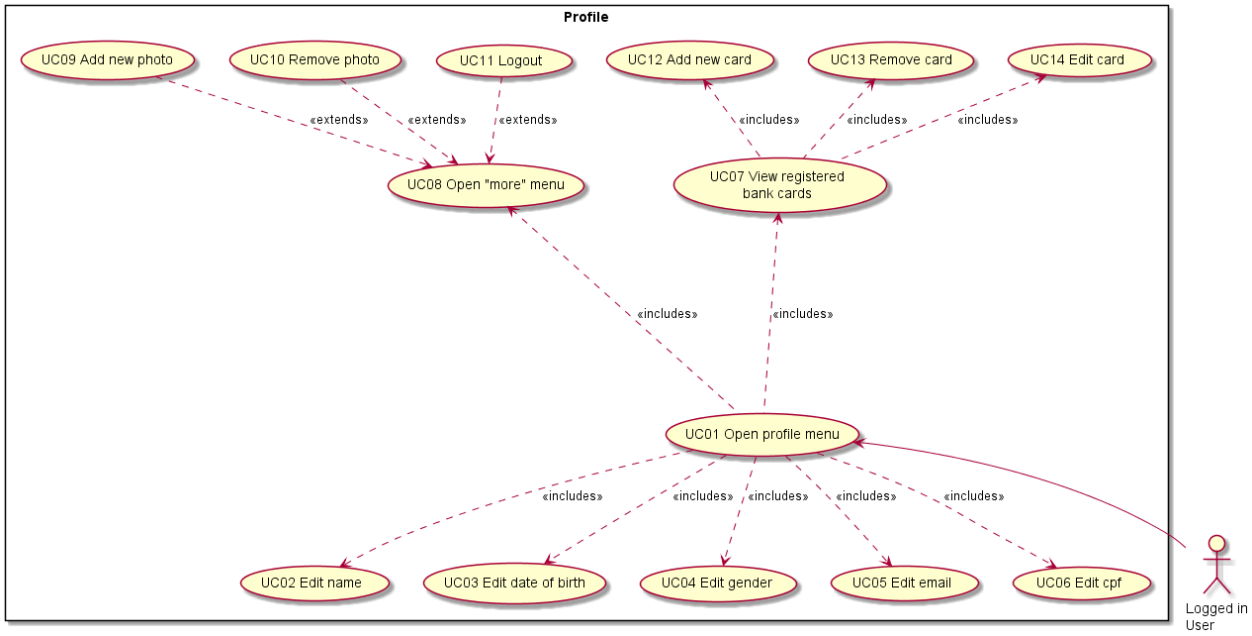



Figura 2 – Diagrama de Caso de Uso - Perfil

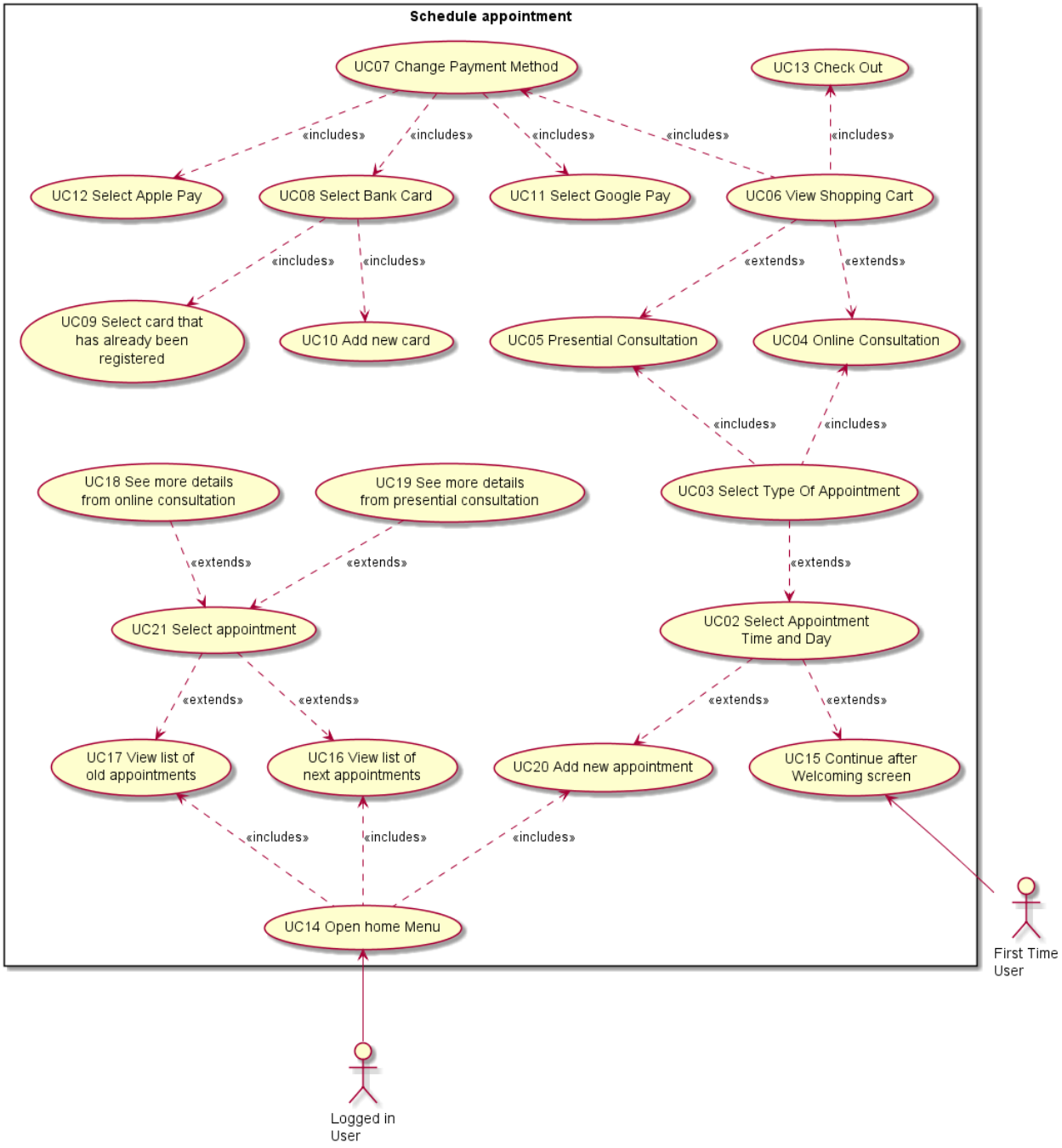



Figura 3 – Diagrama de Caso de Uso - Agendamento de Consulta

### 5.2.3 Bancos de Dados

Da mesma forma, para que fosse concebido os passos iniciais para desenvolvimento do banco de dados, começamos pela criação de um Modelo de Entidade e Relacionamento (MER), então prosseguimos com os Modelos Conceitual e Lógico (Figura 4); evidenciando, então, quais seriam as entidades e seus respectivos atributos/relacionamentos (já que optamos pela

utilização de um banco de dados relacional). Tendo em mãos essas especificações, pudemos avançar com a criação do banco e inserção de dados fictícios, para no futuro poder testar a aplicação.

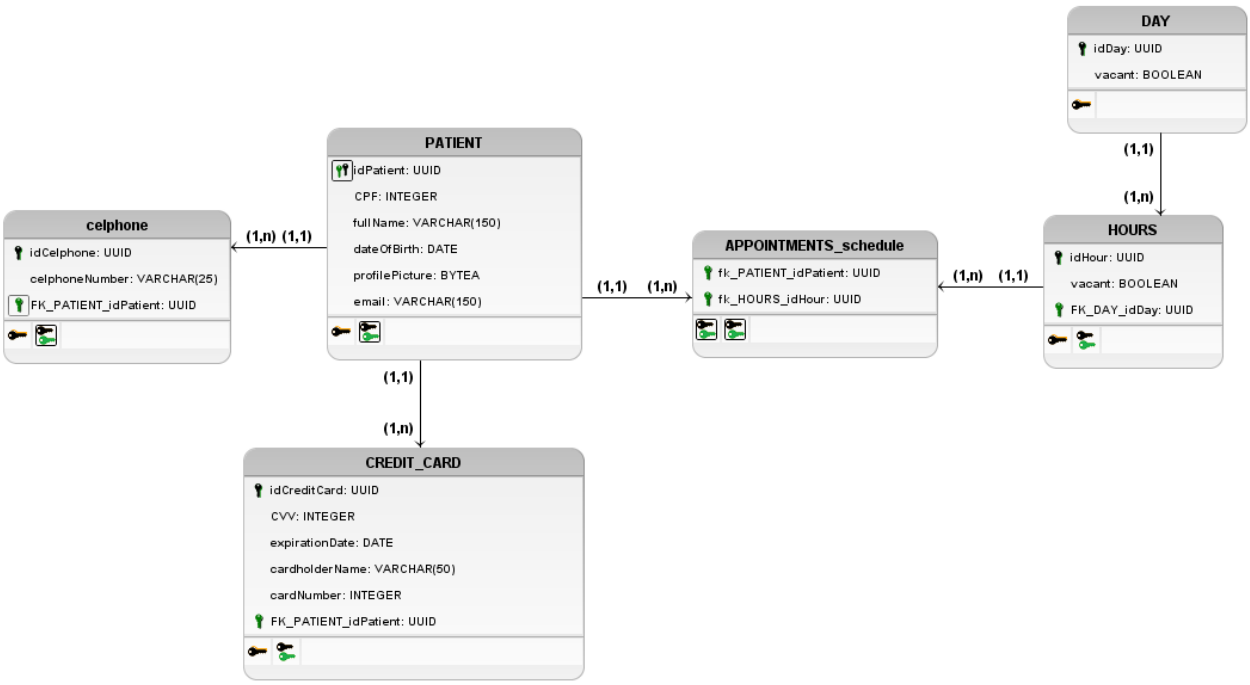



Figura 4 – Banco de Dados - Modelo lógico

### 5.2.4 Definição dos Serviços

Tendo realizado todo aparato para que fosse levantado vários requisitos, partimos para definição de ferramentas.

Por mais que seja um projeto pequeno procuramos trazer para o livro o máximo de Serviços possível, cada qual destinado a um objetivo próprio; como, por exemplo, um serviço para autenticação de usuário, um para controle dos horários disponíveis e um para transações (compras).

Isso foi feito com o intuito de desmistificar o receio de se utilizar tecnologias diferentes das quais o leitor possa estar habituado.

Por fim, os serviços definidos foram:

Tabela 13 – Serviços Estabelecidos

| Setor | Serviços | Tecnlogia | Responsabilidade |
|---|---|---|---|
| *Frontend* | *aplicativo Administrador* | Flutter | aplicativo que admin utilizará para gerenciar seu negócio (serviços para esse aplicativo ainda a ser definido) |
| *Frontend* | *aplicativo Usuário* | React Native (Typescript) | aplicativo que usuário utilizará para marcar consultas |
| *Backend* | *Gateway* | node, graphQL | *gateway* para direcionar requests as devidas APIs |
| *Backend* | *Payment* | a definir, spree | lidar com cartões, pagamentos etc |
| *Backend* | *Auth* | Golang (gorilla/mux, negroni, redigo), Clojure | Autenticação do usuário |
| *Backend* | *Appointment* | node (express, sequelize) | verificar disponibilidade de horários/dias, agendar consultas, log de consultas realizadas |

Isso pode ser melhor visualizado no diagrama abaixo.

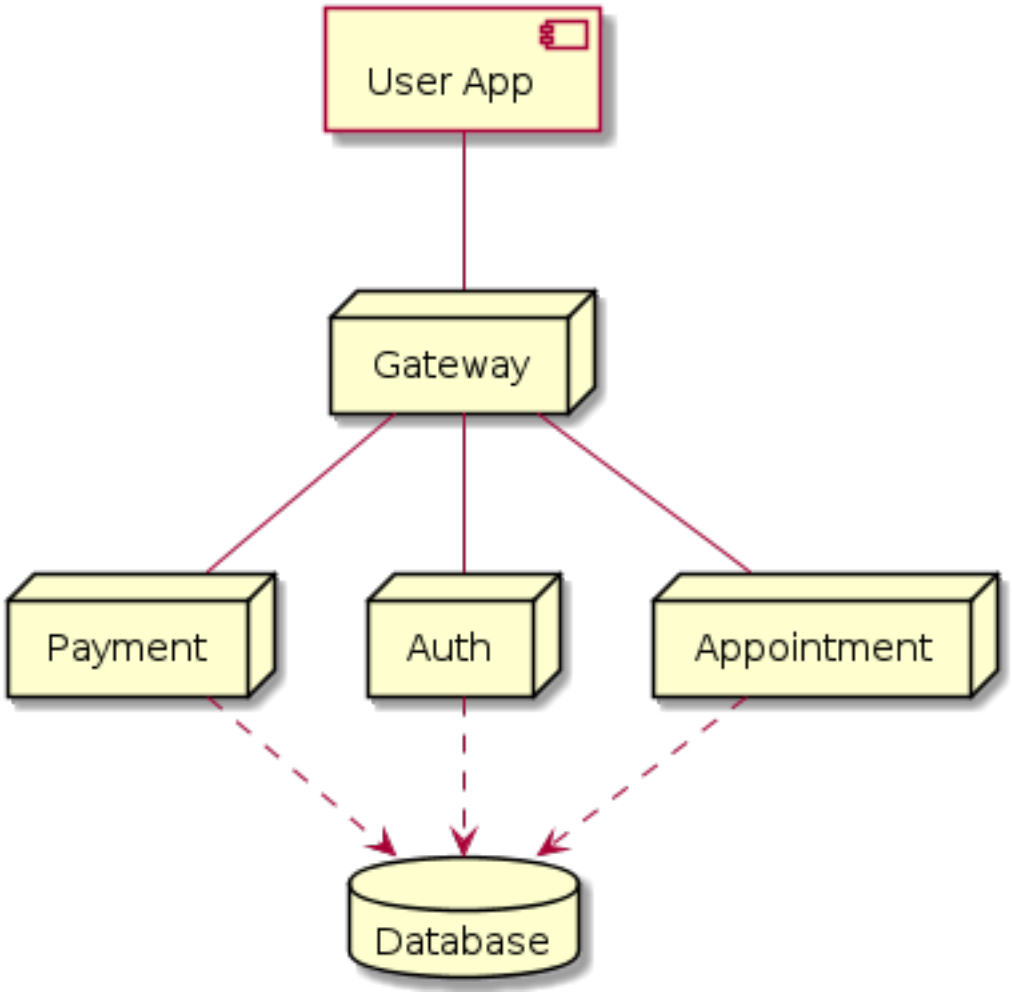



Figura 5 – Microsserviços

#### 5.2.4.1 Microsserviços Desenvolvidos

Os serviços que possuem alguma desenvoltura, nesse momento, são:

***Appointments***:

Microsserviço responsável por todo o CRUD (sigla em inglês para *Criar*, *Ler*, *Atualizar* e *Deletar* ) das consultas. É possível ver as consultas de um usuário (tanto as que passaram, como as que ainda acontecerão) e, também, listar horários disponíveis para agendar.

***User App***:

Estrutura do aplicativo definida, inspirando-se no Atomic Design para estruturação das pastas e componentes; ou seja, os componentes são divididos entre Átomos (unidade básica, reutilizada em vários outros componentes), Moléculas (compostas de átomos), Organismos (compostos de moléculas e átomos). Ainda há, também, Templates (similar a wireframes,

que definirá o layout) e Páginas, que se utiliza de todos os elementos anteriores (Frost 2016). Acrescentamos, também, o uso da ferramenta Storybook, que serve como um *sandbox* (plataforma de testes onde criações e alterações são feitas sem interferir no meio de produção.) para desenvolvimento e visualização de cada elemento da interface de usuário, de forma que estes ficam listados individualmente, facilitando bastante o rápido desenvolvimento e *debug* (tanto em questões de erros de desenvolvimento, como em questões funcionais de responsividade/reatividade, por exemplo).

Algumas telas do fluxo principal foram desenvolvidas; entretanto, até o momento, não há um sistema de rotas entre as mesmas. Além disso, não há interação com as apis.

***Auth***:

Serviço está plenamente desenvolvido, possuindo:

- Rotas de Login, Logout, Signup
- Validação de dados inseridos
- Validação de tokens (jwt)
- Testes unitários
- Testes de integração (api)
- Banco de Dados (PostgreSQL e Redis), usando ORM (Object-relational mapping, "mapeamento objeto relacional que permite fazer uma relação dos objetos com os dados que os mesmos representam" ("Conceitos Básicos em ORM" 2011))
- Orquestração dos containers (com docker-compose)
- Integração contínua, com CircleCi
- Monitoramento da qualidade do código (incluindo cobertura de testes) utilizando SonarCloud

***Oauth2***:

Serviço adaptado para autenticação através do Github. Feito para maior entendimento de como encaixar isso no projeto, portanto foi feito de forma completa (*fullstack*, servindo o *frontend* estaticamente), para que fosse possível simular uma interação. Desenvolvido em Clojure, utilizando a lib Ring.

# 6 Resultados

## 6.1 Questionário

O *survey* recebeu um número significativo de respostas, mais do que o esperado inicialmente. Foram enviados 204 emails e, portanto, julgamos que uma quantia de 20 respostas (~10%) já seria bastante satisfatória. Entretanto, obtivemos 44 respostas, umas mais elaboradas do que outras, mas todas bastante relevantes. Para solicitarmos a participação, utilizamos um template de texto (Seção 8.2).

Pode-se verificar uma visão geral das respostas nesse link.

Em suma, avaliando respostas atreladas às áreas de interesse juntamente com a área de atuação, nota-se que a grande maioria optou por trabalhar em uma empresa privada (36%) ou em uma *StartUp* (25%), atuando como desenvolvedor ou um líder técnico no desenvolvimento.

Junto a isso, a vivência em matérias, que em sua ementa se propunham a ilustrar um projeto real, impactaram de forma positiva na maioria dos entrevistados (52,3% nota 5 e 36,4% nota 4). Isso deve-se principalmente pelo trabalho colaborativo que se tinha como um grupo, como, também, o contato inicial com tecnologias novas ou ferramentas/conceitos que eram utilizados atualmente (Git, gitlab, ruby, SOLID, Kanban, AWS, Cloud...).

Embora seja fato que algumas dessas disciplinas tentam se aproximar ao máximo da realidade, certos assuntos não são abordados, ou são abordados de forma superficial.

Consoante a isso, foi perguntado quais as tecnologias mais utilizadas por cada um atualmente. A lista é grande; e, se pensarmos no escopo de uma disciplina, se torna impraticável conseguir ensinar metade da lista que disponibilizaram. Dessa forma, há de se perceber que, em algum momento, os entrevistados tiveram de ter autonomia suficiente para conseguir aprender tecnologias que viessem a ser mais utilizadas em seu contexto.

Essas ponderações e observações, evidenciaram que a universidade auxilia e muito na formação destes profissionais. Entretanto, grande parte se deve também ao mérito de cada um, que em algum momento tiveram de se aperfeiçoar em algum contexto.

## 6.2 *Ebook*

Novamente, o mesmo pode ser acessado em appointment-octopus.github.io/ebook.

Até o primeiro momento o *ebook* se encontra relativamente em uma fase média: tem uma boa quantidade de conteúdo acerca de *Git*, Linux e Ambientação; além disso, o aplicativo de exemplo já fora abordado no *ebook*, embora há de se ter uma melhor explicação acerca do código em si (e não só do funcionamento superficial dos serviços).

### 6.2.1 Estrutura do *Ebook*

Para que o ebook se comportasse de maneira pragmática, a partir de impressões de Mortimer Adler (2.3.1), estipulamos que os capítulos seriam estruturados da seguinte forma:

- Sumário
- "Resumão" (muito pragmático, o que for mais essencial)
- Explicação (mais conceitual)
- [extra, onde for cabível] FAQ (construído de forma semi-automática)

Para cada grande tópico (por exemplo, *Git*), decidimos que haveria um capítulo de "Resumão"; este teria o propósito de ser o mais prático possível. Dessa forma, o capítulo serviria tanto para antecipar e pré-preparar o leitor que desconhece os conceitos a serem apresentados, como, também, serve para o leitor experienciado como referência para relembrar-se de algum tópico ou questão específica.

Congruente a isso, passamos a escrever cada capítulo do *ebook* seguindo um padrão pré-definido, cujo qual se define em:

- uma estimativa de tempo para leitura daquele tópico
- o que se espera ensinar;
- um parágrafo de introdução que explica os conhecimentos que serão apresentados;
- o conteúdo em si, estruturado em sessões

Além disso, a fim de tornar a leitura mais atrativa, palpável e interessante, decidimos que deveria ser listado um grupo de tópicos, que representaria um conjunto de habilidades cujo as quais propomos fornecer aquele leitor através daquele respectivo capítulo. Fora todas essas questões mais técnicas ou estruturais, optamos por adotar uma linguagem mais coloquial de forma a se tornar uma leitura mais prazerosa, simples e fluida.

## 6.3 Aplicação

Vários passos foram dados e o projeto se encontra bem estabelecido, com boa documentação e os serviços estão evoluídos, restando pouca coisa a ser implementada.

O banco de dados está bem estruturado, o serviço de autenticação e de gerenciamento de consultas se comunicam bem, e, além disso, ambos possuem bastante testes.

No entanto, falta existir uma comunicação entre o *frontend* e o *backend*.

A estrutura de pastas dos serviços se encontra a seguir:

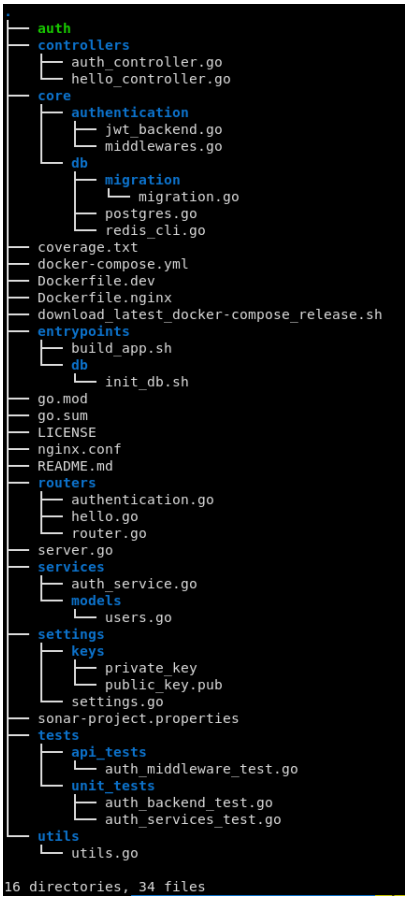



Figura 6 – Auth API

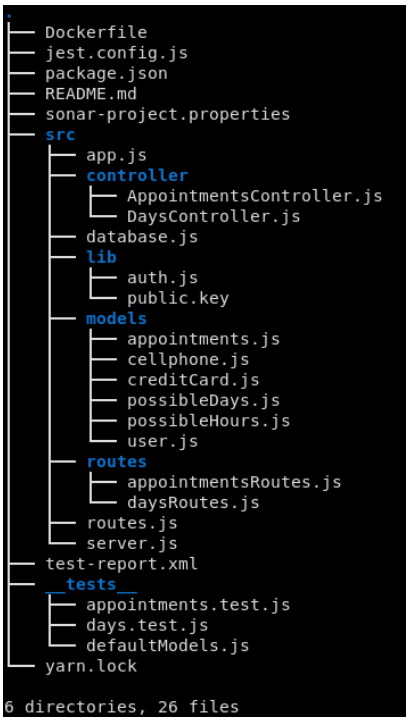



Figura 7 – Appointments API


Figura 8 – Aplicativo do Usuário

# 7 Conclusão

Optar pelo uso de uma literatura cinzenta para suprir a necessidade de uma informação rápida e atual, principalmente para estruturar um guia se tornou uma decisão bastante acertada. Além disso, utilizar princípios de *Software* livre para tornar o *ebook* escalável pela comunidade faz com que ele tenha o potencial de manter-se sempre atual.

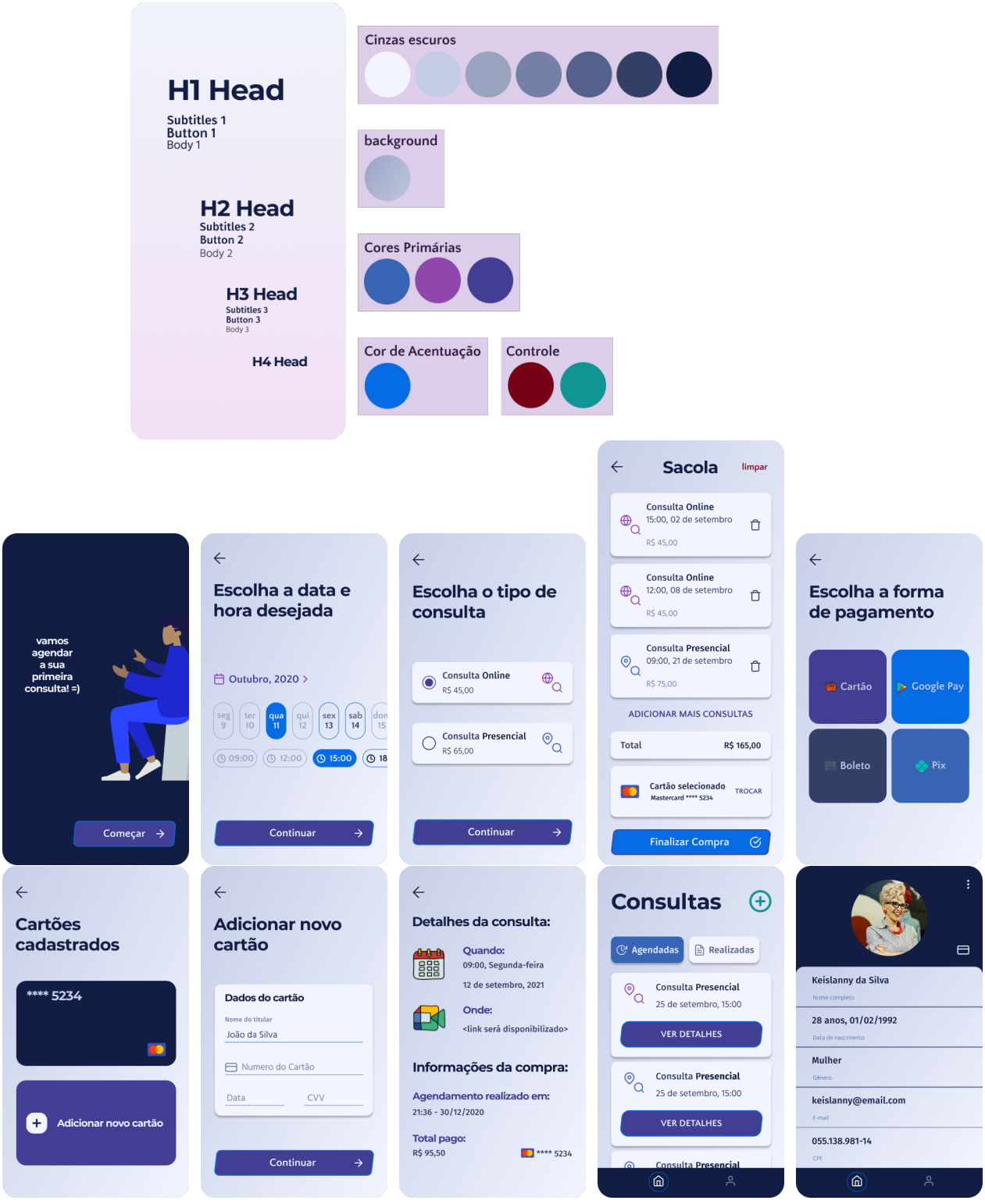
Tendo como base a opinião colhida dentro do questionário, pode-se perceber a falta que faz um material didático, que por sua vez consiga complementar e auxiliar o contexto prático dentro da faculdade. Assim, o *Ebook* pode ser uma referência ao longo de uma disciplina, ou pode ser utilizado por um estudante fora do horário de aula.

A expectativa é que seja um material rico, em português e de fácil acesso para a comunidade de software brasileira. Posteriormente, dependendo do nível de aceitabilidade da comunidade, o mesmo receberia uma versão para inglês mais por conta de visibilidade, para que a adesão ao material pudesse ser maior.

Ainda que a intenção seja abordar o máximo de conteúdo envolvendo desenvolvimento de *software*, o *ebook* permanece, de certa forma, dando passos iniciais dentro de um escopo fechado e teoricamente simples; isto é, os conteúdos presentes podem enriquecer bastante aqueles que nunca utilizaram um Git ou um Linux, nunca fizeram uma API, *dockerizaram* uma aplicação (empacotar um aplicativo em um container Docker) etc. Todavia, existem vários assuntos que podem ser explorados, e grande parte desses foram comentados nas tabelas de tecnologias contidas na Seção 4.1.

# 8 Anexo

## 8.1 Anexo 1 - Prototipo e Tipografia



## 8.2 Anexo 2 - Template email enviado

**Bloco de Código 0.1** Template de email enviado a ex-alunos.

```
<body>
<p>Olá, XXX! Tudo bem? =)</p>
<br/>
<b>Contexto:</b>
<p>Sou estudante de Engenharia de Software pela Universidade de Brasília e,
junto de meu amigo <span style="color: #270d42">Max Henrique</span>, e
orientados pela professora <span style="color: #270d42">Carla Rocha</span>,
estamos compilando um "Guia prático de Engenharia de Software", que tem como
objetivo de guiar desenvolvedores a criar "produto de mundo real" -- isto é,
tocando desde assuntos de elicitação de requisitos, documentação etc até o
deploy em produção.</p>
<br/>
<b>Resumindo:</b>
<p>Criamos um formulário em que pretendemos coletar informações de ex-alunos
sobre as discrepâncias que observaram entre as práticas do Mercado de Trabalho
e o que de fato é ensinado/abordado na Universidade (nas matérias de projeto). </p>
<p>Caso possa dedicar 8 minutinhos para respondê-lo, seria de extrema ajuda.</p>
<p>Segue, enfim, o link (caso possa/queira apoiar-nos):</p>
<a href="https://docs.google.com/forms/d/18tDTBMzFMRs5suLyHAwKD6P9HYdDXl4mvwnh4QUmHqs">
    https://docs.google.com/forms/d/18tDTBMzFMRs5suLyHAwKD6P9HYdDXl4mvwnh4QUmHqs
</a>
<br/>
<p><i>** Vale ressaltar que todos os dados são anonimizados :D **</i></p>
<br/>
<p>Ah! caso queira entrar em contato, por favor não hesite, logo abaixo segue meu
contato e do Max</p>
<p>Telegram: <a href="https://t.me/guilhermedlp">@guilhermedlp</a>,
<a href="https://t.me/ThisMax">@ThisMax</a></p>
<br/>
<p>De antemão, fica aqui nosso agradecimento!</p>
</body>
<footer>
Guilherme de Lyra<br/>
Estudante de Engenharia de Software<br/>
Universidade de Brasília (UnB)<br/>
</footer>
```

# Referências


Adler, Mortimer Jerome, e Charles Van Doren. 1972. "How to Read a Book". *Simon and Schuster*.

Botelho, Rafael Guimarães, e Cristina da Cruz Oliveira. 2017. "Literaturas branca e cinzenta: uma revisão conceitual". *Ciência da Informação*. <http://revista.ibict.br/ciinf/article/view/1804>.

Carneiro, Joana. 2020. "My Software Engineer Roadmap". Medium. novembro de 2020. <https://medium.com/swlh/my-software-engineer-roadmap-2fb0c02b8a08>.

Chakon, Ofir. 2017. "I Switched from Windows to Linux. Here Are the Lessons I Learned Along the Way." Medium. setembro de 2017. <https://medium.com/free-code-camp/i-switched-from-windows-to-linux-here-are-the-lessons-i-learned-along-the-way-434da84ab63f>.

*Como Conseguir Melhores Resultados de Pesquisa | Seja Um Profissional Em Pesquisas No Google*. [s.d.]. Youtube. <https://www.youtube.com/watch?v=3D3U0qgkGvI>.

"Conceitos Básicos em ORM". 2011. 2011. <https://www.devmedia.com.br/orm-object-relational-r
19056>.

"Conduit - RealWorld example App". 2021. maio de 2021. <https://github.com/gothinkster/realworld>.

Filho, Wilson de Pádua Paula. 2000. *Engenharia de Software: fundamentos, métodos e padrões. Proceedings of the conference on {The} future of {Software} engineering*. Editora LTC. <http://aulasprof.6te.net/Arquivos_Aulas/07-Proces_Desen_Soft/Livro_Eng_Soft_Fund_Met_Padroes.pdf>.

Frost, Brad. 2016. *Atomic Design*. 1º ed. <https://atomicdesign.bradfrost.com/>.

Fuggetta, Alfonso. 2000. *Software Process: A Roadmap. Proceedings of the Conference on {the} Future of {Software} Engineering*. ACM Press. <https://doi.org/10.1145/336512.336521>.

Hora, Andre. 2021. "Googling for Software Development: What Developers Search for and What They Find". In. Belo Horizonte, Brasil. <https://elib.dlr.de/139030/>.

Oram, Andrew, e Greg Wilson. 2011. *Making Software: What Really Works, and Why We Believe It*. O'Reilly. <http://portal.acm.org/citation.cfm?doid=336512.336521>.

Wen, Melissa, Leonardo Leite, Fabio Kon, e Paulo Meirelles. 2020. "Understanding FLOSS Through Community Publications: Strategies for Grey Literature Review". In. Seoul South Korea: ACM. <https://doi.org/10.1145/3377816.3381729>.