INSTITUTO DE MATEMÁTICA E ESTATÍSTICA
UNIVERSIDADE DE SÃO PAULO
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

IGGOR FRANCIS NUMATA MATHEWS
LUIS HIKARU SAITO SILVA

EVIDENT APP - A MODERNIZAÇÃO DE UM PROJETO ORIENTADA A BOAS PRÁTICAS DE DESENVOLVIMENTO DE SOFTWARE

SÃO PAULO
24 DE NOVEMBRO DE 2021

IGGOR FRANCIS NUMATA MATHEWS
LUIS HIKARU SAITO SILVA

EVIDENT APP - A MODERNIZAÇÃO DE UM PROJETO ORIENTADA A BOAS PRÁTICAS DE DESENVOLVIMENTO DE SOFTWARE

Trabalho de Conclusão de Curso apresentado ao curso de Ciência da Computação da Universidade de São Paulo, como monografia da disciplina MAC0449 - Trabalho de Formatura Supervisionado e como parte dos requisitos necessários à obtenção do título de Bacharel em Ciência da Computação.

Orientador: Prof. Dr. Alfredo Goldman vel Lejbman

Co-orientador: BsC. João Francisco Lino Daniel

SÃO PAULO

2021

Dedicamos este trabalho a nossa família que nos acompanhou até aqui.

Agradecimentos

Ambos:

Nós, Iggor e Luis, temos muito a agradecer aos professores, Alfredo Goldman e João Lino Gabriel, à nossa cliente Gabriela Machado, bem como à equipe EviDent, por todo o apoio e trabalho conjunto que empregamos para a realização deste projeto. Foram incontáveis horas de desenvolvimento e reuniões, mas sempre foi com muito bom espírito, companheirismo e alegria que tínhamos nossas reuniões quinzenais.

Hikaru:

Gostaria de agradecer à minha mãe, que nunca deixou de me acompanhar e apoiar durante essa jornada, e ao Iggor, por tudo que passamos juntos durante a graduação.

Iggor:

Gostaria de agradecer a todos os amigos que acompanharam esta jornada por fora, pessoas que estiveram do nosso lado desde o início e principalmente às que ainda comemoram conosco essa realização. Também à minha família, que foi muito compreensiva durante um ano que recusei tantas vezes passar um tempo de qualidade com pessoas tão queridas e que sempre me apoiaram para que eu perseverasse, principalmente quando mais precisei.

Um agradecimento especial a todos os colegas e amigos que conheci no período do primeiro estágio, atual local de trabalho, aos colegas da faculdade e professores que me apoiaram mesmo nas adversidades e infortúnios da graduação, pois são essas pessoas que me permitiram prosperar e aprender a ponto de produzir algo tão gratificante quanto este projeto.

Não somente um agradecimento, mas também um reconhecimento de todo o trabalho que tive com o Luis neste último ano, já não aguentávamos mais nos encontrar e estar sempre juntos, mas tenho certeza de que essa pessoa foi a melhor companhia para todas essas batalhas, a real amizade que nunca esquecerei.

E, por fim, o mais especial de todos os agradecimentos, à Vitória Chiovetto, Vica, minha namorada, meu "Chuchu", que sempre acreditou, suportou e festejou comigo, nunca importando a ocasião, e durante todos os dias me elevou para um humor melhor e confiança na minha capacidade, uma companheira de verdade e sem fim.

Resumo

Este Trabalho de Conclusão de curso consiste no desenvolvimento de uma aplicação Web, cuja concepção criativa foi originalmente feita pela doutoranda Gabriela Machado. Nele, exploramos as vantagens da aplicação de metodologias ágeis durante um período extenso de desenvolvimento de software, assim como o uso de tecnologias comuns para a criação de sites no mercado atual. Isso foi feito com o objetivo de ter como resultado final uma plataforma de disseminação de conhecimento odontológico baseado em evidência científica de fácil acesso para profissionais que atuam na área e estabelecer uma documentação adequada para a continuidade do progresso em anos seguintes. Com esse objetivo em mente, o projeto final consiste em duas aplicações: *frontend* e *backend*, alimentadas por um *Content Management System* que pode ser utilizado por pessoas sem conhecimento na área de programação. Ambas as aplicações seguem uma arquitetura dividida em camadas, cujos benefícios no processo de desenvolvimento foram explorados assim como as metodologias ágeis.

Abstract

This Capstone Project consists on developing a Web application, which was conceptually conceived by doctoral student Gabriela Machado. In it, we explore the advantages of using agile methodologies in an extended software development period, as well as the use of common benchmark technologies to create websites in today's current tech landscape. This project has the objective of creating an accessible platform for disseminating dental knowledge based on scientific evidence to the field's professionals, as well as creating an adequate documentation for the project's continued development in upcoming years. With this in mind, the final work consists of two applications: frontend and backend, fed by a Content Management System that can be operated by people with no programming knowledge whatsoever. Both applications possess a layer-based architecture, whose benefits in the development process were explored and documented similarly to agile methodologies.

Sumário

1. Introdução

Atualmente, o EviDent é uma aplicação Web com porte para dispositivos móveis, que visa ser uma plataforma para profissionais da odontologia receberem conhecimento digerido de novas evidências científicas e gradualmente se tornarem autônomos no processo de decisão clínica baseado em evidências científicas. Originalmente a iniciativa estava apenas em uma página no Facebook, hoje em dia a aplicação visa alcançar profissionais da área de odontologia e ser usada em sua rotina profissional.

O projeto nasceu na Faculdade de Odontologia da Universidade de São Paulo, motivado pela dificuldade dos profissionais em buscar e avaliar criticamente as evidências científicas disponíveis na área da odontologia para aplicá-las em sua prática. Assim sendo, foi criada uma página no Facebook para este fim e após um ano de acompanhamento foi notada falta de engajamento e passividade dos usuários. Assim, decidiu-se criar uma aplicação para que fosse possível veicular diferentes formatos de conteúdo e possibilitar maior interação dos usuários.

Conhecendo o professor Goldman, os alunos responsáveis pelo EviDent decidiram trazer a proposta da aplicação para que fosse objeto de estudo na matéria **MAC0472 - Laboratório de Métodos Ágeis** (também conhecida como LabXP), e foi criada a primeira versão do EviDent, utilizando um framework de JavaScript chamado Ionic. Durante a distribuição de 2020 da matéria de *LabXP* que foi oferecida pelo professor Goldman, os alunos Iggor Numata e Luis Hikaru tiveram seu primeiro contato com o projeto, de forma que houve muitas dificuldades no desenvolvimento, parte pela falta de documentação deixada nas distribuições anteriores da disciplina, e parte pela decisão de usar o Ionic para o desenvolvimento da aplicação.

Sendo um framework de JavaScript com suporte para código nativo, fazia sentido utilizá-lo na concepção do projeto, porém, durante os anos de desenvolvimento que se passaram, a comunidade de desenvolvimento *mobile* deixou de usar a tecnologia, tendendo a novos frameworks e ao uso de PWAs (*Powered Web Application)* para simular aplicações nativas com código Web. Dessa

forma, havia pouquíssimo suporte online para resolver os problemas da plataforma, ainda mais se tratando de uma versão mais antiga do Ionic.

Além disso, se tratando de uma aplicação nativa, o projeto necessitava de hardware e software da Apple para lançamento de novas versões na AppStore (loja virtual de software para dispositivos iOS), que não era acessível a todos os membros do grupo. Refletindo sobre todos esses problemas, e a notória falta de escalabilidade de uma aplicação em Ionic, foi feita, em conjunto, a decisão de refazer a aplicação com outras tecnologias, com o argumento de que, a longo prazo, seria uma mudança positiva para o EviDent como um todo. Assim decidiu-se criar uma nova aplicação, utilizando tecnologias com comunidades mais ativas e de fácil manutenção.Após a conclusão da matéria, os alunos decidiram dar continuidade ao projeto como seu Trabalho de Conclusão de Curso.

Os objetivos estabelecidos foram de re-criar o EviDent em uma nova plataforma, mantendo boa usabilidade de usuário, e ao mesmo tempo seguir boas práticas do desenvolvimento ágil para criar uma aplicação bem documentada e com arquitetura definida, de forma que fosse escalável e tivesse código limpo e legível. Tais conceitos são explorados em *Clean Code: A Handbook of Agile Software Craftsmanship,* e foram aplicados aqui.

Esta monografia foi dividida em sete capítulos, sendo esta Introdução o primeiro deles. O capítulo dois apresenta um relato da Gabriela Machado em relação aos objetivos iniciais e trajetória do projeto até o momento atual. O capítulo três explora a aplicação desenvolvida, expondo suas funcionalidades e divisões de responsabilidade entre *frontend* e *backend,* de forma que o capítulo quatro entra em mais detalhes sobre as tecnologias e metodologias usadas para o desenvolvimento da aplicação, desde *frameworks* até a estrutura de *Gitflow* utilizada. O capítulo cinco explora possíveis melhorias e novas funcionalidades que poderiam ser adicionadas com mais tempo de desenvolvimento. O capítulo seis contém um relato pessoal de cada aluno avaliado e sua atuação individual no projeto, e, por fim, o capítulo sete contém as conclusões de cada aluno, assim como um pequeno relato final da Gabriela Machado.

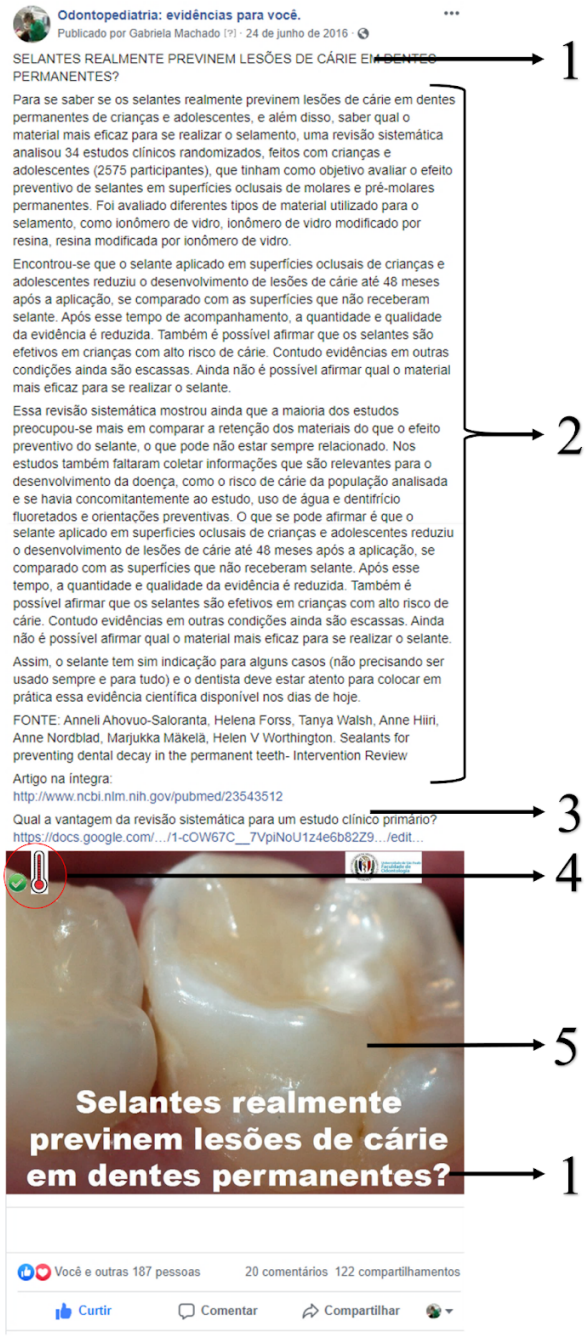
2. EviDent

O objetivo deste capítulo é apresentar um relato escrito pela Gabriela Machado, dona original da propriedade criativa do *EviDent*, que atuou como *product manager* durante o desenvolvimento. Dessa forma, podemos ter uma ideia mais apurada da intenção original da criação da plataforma, assim como sua jornada antes do início do desenvolvimento pelos alunos avaliados. *EviDent* é um projeto amplo de pesquisa que nasceu na Faculdade de Odontologia da Universidade de São Paulo. Ele visa promover melhores condições para o dentista exercer a prática odontológica baseada em evidências no Brasil, melhorando a captação de evidências científicas para embasar as decisões dos profissionais e, consequentemente, melhorar os sistemas e serviços de saúde. Além de disseminar o conhecimento traduzido de evidências científicas, as iniciativas desse projeto também se concentram em capacitar gradualmente os dentistas para uma autonomia na tomada de decisão, independente de recomendações de terceiros.

O grupo de pesquisa do departamento de Ortodontia e Odontopediatria da FO USP já se preocupava em delinear estudos visando fornecer informações relevantes para a prática clínica, que pudessem ser prontamente aplicados pelos profissionais. No entanto, os profissionais enfrentam alguns obstáculos em relação ao acesso e interação com artigos científicos e iniciativas que visam facilitar essa interação eram necessárias. Assim, o *EviDent* nasceu em 2013 por meio de uma página do Facebook intitulada "Odontopediatria: Evidências para você!”, em que postagens com imagens ilustrativas e títulos em formato de perguntas introduziram um texto curto apresentando os achados de estudos científicos relevantes para a prática clínica selecionados por especialistas (Figura 1).

No entanto, após um ciclo piloto observamos a passividade dos usuários em relação ao acesso do artigo, disponível por meio de um link e interação com a postagem, sendo a imagem a parte que mais recebeu algum tipo de interação. Assim o grupo passou a trabalhar com formatos gráficos de apresentação do conteúdo e outras redes sociais como o Instagram (*EviDent)*. Além disso, seguindo a tendência de mercado e visando prover maiores oportunidades de interação ao usuário, bem como concentrar as informações em um só lugar em que o clínico

pudesse acessar em seu dia a dia, o desenvolvimento de uma aplicação se fez necessária.



Nº		Descrição
1	**Título**	Escrito em formato de pergunta para chamar a atenção do leitor
2	**Conteúdo do artigo "Digerido"**	Texto mais curto o possível,escrito com linguagem focada nos dentistas.
3	**Links**	Cada tópico apresentava 2 links: link para o artigo principal (3a) e link para uma explicação sobre o desenho do estudo e sua relevância para a prática clínica e o processo de tomada de decisão (3b).
4	**Símbolos**	Os símbolos são carimbados na página de rosto da postagem para indicar a relevância da evidência digerida (classificação do desenho do estudo e julgamento da certeza da evidência disponível em relação a outros desenhos possíveis a serem usados ou limitações a serem superadas em estudos posteriores).
5	**Figura**	Imagem relacionada ao tema

Figura 1 - Estrutura de postagem sistematizada do Facebook

Acreditamos que por meio do aplicativo o dentista consiga acessar as informações que ele deseje de forma mais rápida do que nas redes sociais, além disso, o profissional pode baixar o conteúdo que achar necessário para ter a disposição mesmo quando não possuir rede de internet disponível. No entanto, em

nosso grupo de dentistas não era possível o desenvolvimento do aplicativo em si, então iniciamos o planejamento dos recursos que gostaríamos por meio de rascunhos e um protótipo, que foi apresentado pela primeira vez em 2018 na disciplina de **Desenvolvimento Ágil de Software com Programação eXtrema (XP)** resultando em uma primeira versão do aplicativo para sistemas operacionais android (Figura 2b) desenvolvida pelos alunos de graduação e pós-graduação do curso de ciências da computação do Instituto de Matemática e Estatística da USP em conjunto com os alunos do projeto da Faculdade de Odontologia. A versão foi registrada em 2020 (Machado et al.) e nos possibilitou testes iniciais com usuários alvo (amostra restrita) que apontou para novas demandas que foram implementadas por meio da mesma disciplina em 2019 (Figura 2c) e 2020 (Figura 2d).

Mesmo com os aprimoramentos realizados, a equipe do ano de 2020 percebeu a necessidade de alterar a tecnologia e o framework usado no desenvolvimento apontando para uma solução tecnológica que foi desenvolvida ao longo do ano de 2021, resultando no aplicativo final que temos atualmente. O *EviDent* app foi lançado em setembro de 2021 em formato de mvp para testes com uma amostra espontânea, ou seja, divulgações foram realizadas em um congresso com potenciais usuários e algumas métricas de uso foram coletadas por meio do Google Analytics. Também criamos um formulário no aplicativo para que o usuário pudesse avaliar a usabilidade e dessa forma tivéssemos dados suficientes para aprimorar o MVP (*Minimum Viable Product)* e desenvolver o aplicativo final.

Sobre o EviDent, a dentista Gabriela Machado que foi a idealizadora do projeto afirmou após a conclusão do projeto:

> Como dentistas, participar do desenvolvimento em todos os ciclos foi uma oportunidade interdisciplinar e enriquecedora. A comunicação para transmitir ideias e entender como um software funciona foi um grande desafio, tanto para o nosso grupo de dentistas quanto para o grupo de desenvolvedores que teve que adaptar a linguagem e construir uma comunicação eficaz. (MACHADO, Gabriela.)

Figura 2 - Screenshots das versões do aplicativo EviDent. A - Rascunho projetado pela equipe de dentistas; B- primeira versão do aplicativo (registrada); C- aprimoramento da primeira versão; D - última versão (PWA).

3. Proposta

Neste capítulo, veremos a divisão de responsabilidades entre *frontend* e *backend,* assim como as funcionalidades individuais presentes em cada um. O projeto é atualmente uma aplicação web dividida em um *frontend* e um *backend*. No *frontend*, há a aplicação com a qual o usuário interage, e, no *backend*, há um serviço responsável por gerenciar os dados no banco de dados. Ambos os repositórios estão armazenados na plataforma **GitLab**[1] em um repositório privado.

3.1 Frontend

Como muitos aplicativos, o EviDent possui um fluxo de cadastro de usuário, voltado principalmente para informar a equipe de desenvolvimento sobre as demográficas de cada usuário, possibilitando assim que, a longo termo, o projeto se adeque ao seu usuário médio, guiando novas funções a serem desenvolvidas e melhorias em funções já existentes. Tais dados de cadastro são salvos no *backend* e geridos através de tokens de identificação JWT (JSON Web Tokens) que ficam salvos nos cookies de navegador do usuário. A principal funcionalidade, entretanto, são as postagens de conhecimento acadêmico odontológico, divididas em diversas categorias para facilitar a navegação do usuário. A plataforma em si apenas consome esses dados, mostrando-os conforme o usuário navega pela aplicação.

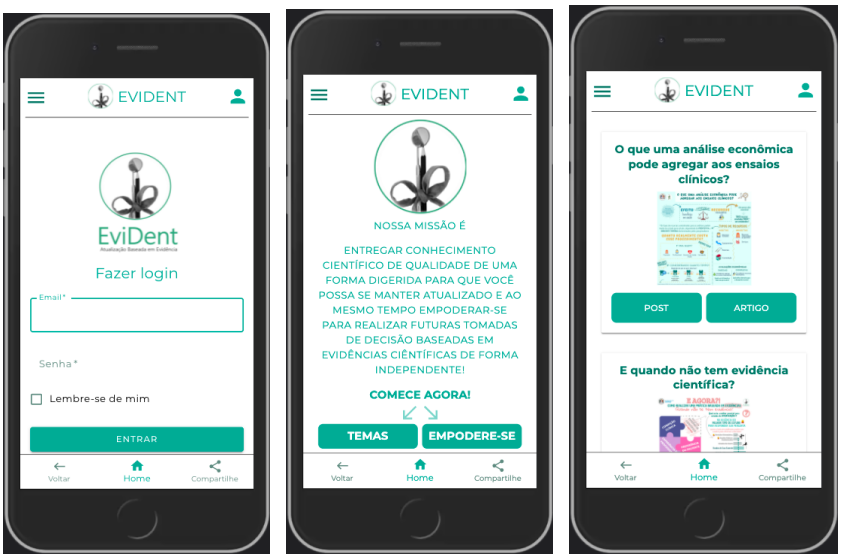


Figura 3 - Prints atuais de diversas partes da aplicação

[1] GitLab. Disponível em: <https://gitlab.com/ebapp> Acesso ao longo de 2021.

Além disso, utilizamos o Google Analytics como uma ferramenta de monitoramento do usuário, a fim de colher dados que possam ser usados para guiar o desenvolvimento futuro.

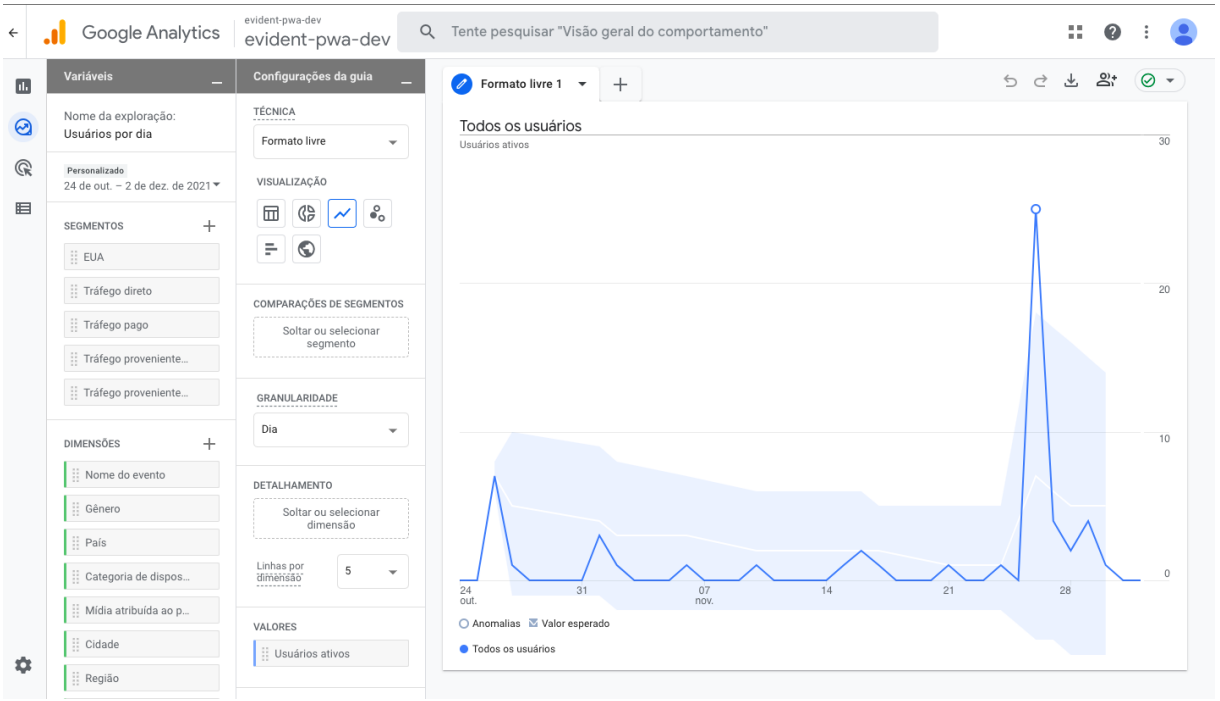


Figura 4 - Painel do Google Analytics apresentando o número de usuários diferentes que usam a aplicação dia a dia.

3.2 Backend

O servidor *backend* do EviDent utiliza de um sistema gerenciador de banco de dados dos mais aceitos no mercado, MongoDB, o mais popular dos bancos NoSQL, para armazenamento e recuperação de dados principais da aplicação. E ainda, é integrado o Contentful com um sistema de gerenciamento de conteúdo (CMS), para maior facilidade e segurança no gerenciamento do principal conteúdo oferecido pela aplicação, as postagens. Ainda, nossa plataforma conta com fluxos completos de cadastro e autenticação de usuários para melhor monitoramento da sessão destes. Bem como validação e tratamento dos dados enviados do *frontent* para o *backend*, antes de serem armazenados no nosso banco de dados. Também, filtragem e gerenciamento de postagens por categorias, que também são gerenciadas por requisições ao nosso servidor.

4. Métodos e Tecnologias

Neste capítulo, vamos explorar o processo de desenvolvimento, quais estratégias foram adotadas e, por fim, daremos uma visão ampla e aprofundada de todas as diferentes tecnologias utilizadas, acompanhando sempre a motivação pela sua escolha. Como o primeiro contato da equipe com o projeto foi através da matéria **MAC0472 - Laboratório de Métodos Ágeis**, foi um acordo unânime seguir algumas diretrizes de Desenvolvimento Ágil, mais especificamente, do *Scrum.* Assim sendo, o projeto em si foi dividido em diversas pequenas entregas que começaram em Março de 2021, seguindo um calendário quinzenal com reuniões de alinhamento constantes para garantir que o desenvolvimento não fosse estagnado.

Enquanto a metodologia original possui diversas cerimônias de equipe para manter alinhamento e reflexão sobre o desenvolvimento, foi concluído que tais cerimônias não seriam adotadas em um primeiro momento, e, como o desenvolvimento teve um bom ritmo, elas nunca foram adotadas pela equipe. Além disso, foi decidido adotar uma divisão de código em camadas tanto no *frontend* quanto no *backend*, a fim de criar um software de fácil legibilidade futura.

4.1.1 Metodologia Ágil - *Scrum*

Sumariamente, a função dessa metodologia de desenvolvimento é dividir a entrega de software em pequenas funcionalidades. Mais importante, ela fornece um ênfase em **comprometimento de entregas**, planejando o desenvolvimento previamente e alinhando tanto as expectativas da cliente quanto do nosso orientador. Acreditamos que a adoção dessa metodologia foi uma das melhores decisões feitas pela equipe, pois mantivemos um ritmo acelerado e dedicação durante todo o período, algo que ficou aparente durante o alinhamento de progresso com a professora e doutora Nina S.T Hirata. Dessa forma, o projeto pode ser visto como um atestado da eficiência da filosofia Ágil mesmo com uma equipe de desenvolvimento pequena.

4.1.2 Desenvolvimento em equipe - Git & Gitflow

O projeto está todo hospedado no gitlab de autoria da equipe de desenvolvimento do EviDent, que pode ser encontrado aqui. O gitlab do EviDent é

privado dada a demanda da Gabriela, porém é relevante apontar que, somados, o *frontend* e o *backend* possuem 148 *commits*, sendo 106 do *front* e 42 do *back*, e em torno de 62MB de arquivos.

Para garantir que não haveriam conflitos entre as alterações dos dois membros do grupo, nós definimos o seguinte *Gitflow* para ambos os repositórios:

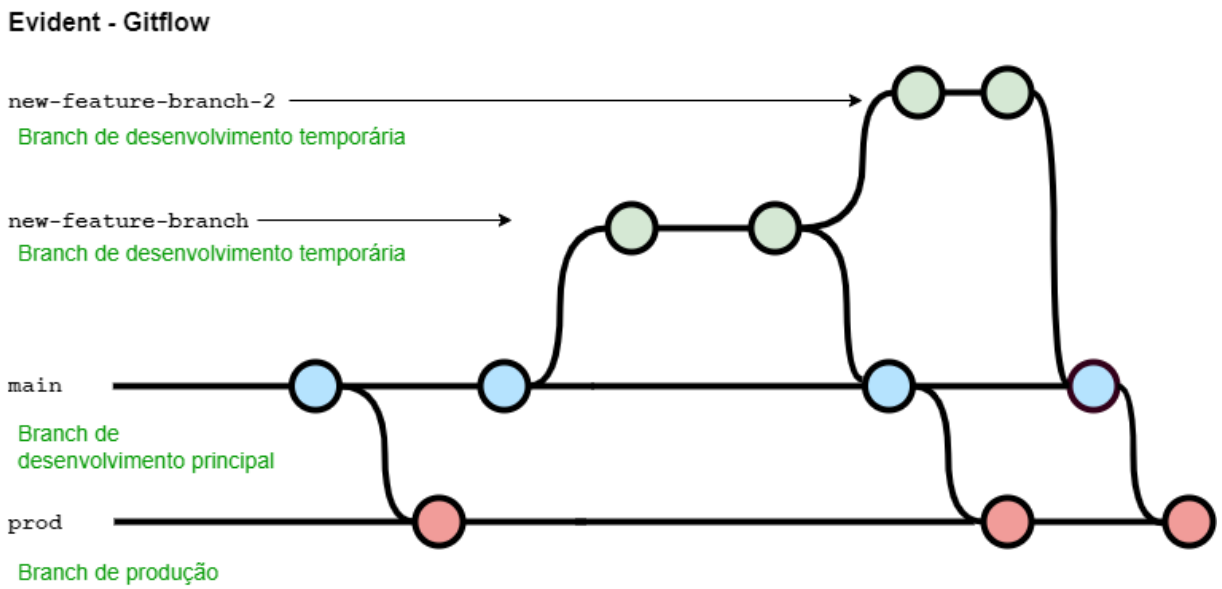


Figura 5 - Diagrama visual do gitflow utilizado

Neste modelo, a *main* é considerada a principal *branch*, e fonte de verdade do repositório. Ao adicionar alterações nela, nós garantimos que essas alterações foram testadas e devem estar presentes na aplicação em produção. Já a *branch prod* é a *branch* de produção. No *frontend,* quaisquer alterações nela geram uma nova versão automaticamente, e portanto deve-se atentar a não inserir alterações defeituosas nessa *branch*. Novas funcionalidades devem ser desenvolvidas a partir da *main,* ou a outra *branch* de funcionalidade.

4.1.3 Organização do código - Clean Code

Um dos principais problemas que encontramos com o projeto antigo em 2020 foi uma notória falta de documentação e organização interna do código. Assim sendo, decidimos estabelecer regras de desenvolvimento fundadas nos conceitos de *Clean Code* para garantir a longevidade e facilidade de manutenção do repositório. Essa decisão não só nos levou a padronizar componentes e elementos relacionados

à arquitetura interna, como também documentar tais decisões nos arquivos README.md encontrados em cada repositório.

4.2 Infraestrutura Geral

Na figura abaixo, podemos ver uma representação visual da infraestrutura do projeto, listando as tecnologias usadas e como elas interagem entre si:

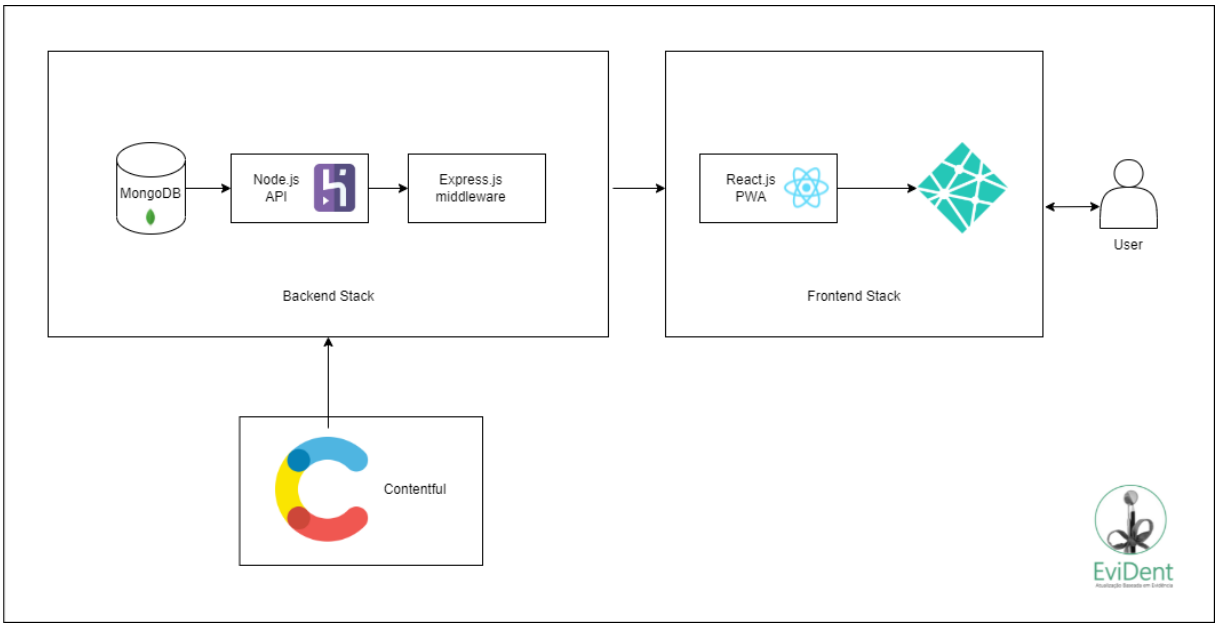


Figura 6 - Arquitetura completa do projeto

Este capítulo fornecerá uma descrição detalhada de cada tecnologia e, naturalmente, um entendimento melhor do diagrama.

4.3 Frontend

4.3.1 Tecnologias Utilizadas

A aplicação foi desenvolvida em React.js, que é um *framework* de desenvolvimento feito com base em Javascript. Apesar de ser uma aplicação web, foi desenvolvido um porte para PWA (*Powered Web Application*) para dar mais conforto ao usuário de dispositivo móvel, que representa a grande maioria dos usuários. Para hospedar a aplicação, foi utilizada a plataforma Netlify, que por sua parte, faz uso de recursos da Amazon. A plataforma foi recomendada pelo Professor

Goldman pela sua comodidade, e assim poupou à equipe a preocupação de gerenciar o serviço de hospedagem.

Para acelerar o desenvolvimento e manter uma melhor legibilidade do código, também foi feito uso de algumas bibliotecas de auxílio, como o Material-UI e o Axios.js. Para manter um monitoramento de navegação do usuário, inserimos também uma integração com o **Google Analytics**, que trouxe dados importantes que podem ser usados para guiar e melhorar o desenvolvimento da plataforma no futuro.

4.3.2 Infraestrutura

A aplicação foi desenvolvida para interagir diretamente com o usuário e estando hospedada em um serviço de nuvem, pode ser acessada tanto por um navegador quanto diretamente pela tela inicial de um dispositivo móvel, se o usuário desejar salvá-la de tal forma. Além disso, é diretamente alimentada por um servidor *backend* (explicado em mais detalhes na próxima seção), e possui uma integração com o Goole Analytics, servindo eventos à plataforma a fim de manter um monitoramento do engajamento dos usuários.

4.3.3 Arquitetura do projeto

Adotamos uma arquitetura simples, mas muito importante para a organização do código. A ideia é dividir componentes semelhantes de forma que estejam no mesmo diretório, e centralizar partes específicas das funcionalidades em camadas intuitivas que auxiliem o diagnóstico e resolução de problemas que possam surgir. Apesar da arquitetura ter sido baseada em conceitos de *Clean Architecture,* a estrutura em si não segue esses conceitos à risca, pela simplicidade do projeto. Todos os diretórios abaixo encontram-se no diretório .src dentro da aplicação de frontend.

- *Components:* Abriga componentes **visuais** que são feitos para uso geral na aplicação. Tais componentes costumeiramente são funções que devolvem elementos HTML a serem inseridos no corpo da aplicação.

- *Constants:* Abriga **textos, números e objetos de mapeamento relacional** que são padronizados pela aplicação de forma que as alterações afetam o visual do projeto como um todo.
- *Global:* Mantém funções para armazenamento e acesso de variáveis dentro do **estado global** da aplicação, ou seja, variáveis que podem ser acessadas em qualquer página do projeto. Atualmente, há pouquíssimos dados que precisam ser propagados entre páginas, e por isso, fazemos uso de Cookies de navegador para tal.
- *Models:* Contém interfaces de tipagem que visam **padronizar modelos de aplicação** no projeto, assim, sendo responsável por **traduzir regras de negócio para interfaces de objeto.**
- *Screens:* Responsável por guardar os elementos **visuais e funcionais** de cada tela da aplicação e, se necessário, cada tela pode possuir um diretório com seus componentes individuais. Essa é a camada que recebe mais interação do usuário.
- *Services:* Implementa funções de **integração com a API do backend**, e realiza busca de dados para popular os componentes visuais.
- *Utils:* Possui funções de utilidade geral que não se enquadram na camada de componentes.

4.4 Backend

4.4.1 Tecnologias Utilizadas

O servidor *backend* foi construído em Node.js. Normalmente, o código JavaScript seria executado em um navegador, com o uso deste *framework*, porém nosso servidor dispensa essa necessidade e pode ser executado na nuvem. Este é o cerne do servidor, ou seja, partindo disto, foram incluídas outras bibliotecas auxiliares para melhor manipulação, gerenciamento e validação dos dados, bem como bibliotecas voltadas a facilitar a experiência de desenvolvimento. Com a finalidade de facilitar o processo de criação de um servidor com uma Interface de Programação de Aplicações (API), foi também utilizada a ferramenta Express.js, um *framework* responsável por configurar demais aspectos do servidor Node, bem como criar e gerenciar as requisições HTTP.

Utilizamos também o MongoDB, um sistema gerenciador de banco de dados NoSQL, orientado a documentos, para armazenar e gerenciar os dados principais da aplicação. Para tanto, fizemos uso do MongoDB Atlas, um servidor *cloud* desenvolvido pela mesma equipe do MongoDB o que facilitou o seu uso e compatibilidade com o nosso banco de dados. Também foi utilizada uma biblioteca para fazer a conexão do express com o banco de dados, o Mongoose. O servidor em si foi implantado na plataforma *cloud* Heroku, grandemente utilizada pelo amplo suporte a diversas linguagens de programação. Diferentemente do *frontend*, que é hospedado no Netlify, agora não poderíamos fazer uso da mesma plataforma, pois ela não suportava um servidor *backend* como planejamos para este projeto.

Há também o Contentful, uma plataforma voltada à criação e gerenciamento de conteúdos e de fácil integração com servidores. A ideia é que ela seja a primeira versão da plataforma de administradores do projeto, por onde será possível introduzir novas postagens. Assim, dando mais liberdade para a cliente que dispensa uma equipe de desenvolvedores para atualização e lançamentos de novas postagens no seu produto.

Por fim, vale ressaltar que, embora todas as principais tecnologias utilizadas neste projeto trabalhem com a linguagem do *JavaScript*, foi preferível durante o tempo de desenvolvimento o uso do TypeScript. Está é uma linguagem construída em cima do *JavaScript*, mas com suporte à tipagem estática dos seus elementos e foi usada para auxiliar em nossa busca por boas práticas de desenvolvimento, visando facilitar o processo de entrega e introdução do projeto para novos colaboradores.

4.4.2 Arquitetura

Semelhante ao adotado pelo *frontend*, a arquitetura aplicada no servidor *backend* não é feita para ser complexa, mas sim para reforçar um bom desenvolvimento do produto, organizar os arquivos e facilitar a leitura e a documentação do projeto seguindo princípios simples. Novamente tentamos focar nossos esforços para seguir a arquitetura proposta por Robert Cecil Martin em seu livro *Código Limpo*, de 2008. Essa proposta descreve o projeto como diversas camadas, ou níveis, em que em cada uma são agrupados arquivos de mesma

função. Essas camadas têm um objetivo muito claro que pode ser resumido em: facilitar a alteração e manutenção do código, ainda que alguma grande mudança seja realizada à nível de infraestrutura. Mais uma vez, os diretórios abaixo encontram-se no diretório **src** , agora dentro do servidor *backend*:

- **_common**: engloba arquivos acessíveis e usados por outros diretórios. Estes arquivos incluem, em sua maioria, constantes, classes, interfaces e funções comuns a mais de um nível da arquitetura ou mais de um arquivo;
- **core/datasources**: representando uma camada de grande importância para os princípios da arquitetura adotada, os *datasources* são apenas interfaces (tipo de estrutura do *TypeScript* utilizado para informar a tipagem de um objeto, sem implementá-lo) que são consumidas por outros arquivos para fazer requisições para o banco de dados por apenas invocar seus métodos. Além disso, como não implementam nada, de fato, são implementados em outra camada;
- **core/useCases**: utilizada para armazenar as regras de negócio da aplicação, gerenciar as requisições de dados para o banco que devem ser feitas e como devem ser tratadas tais respostas. Em geral, é a camada que mais se preocupará com a lógica e fluxo dos dados, bem como, implementar quaisquer algoritmos e estruturas para viabilizar os comportamentos esperados de uma chamada para a *API*;
- **database**: responsável por fazer a maior parte das comunicações com bancos de dados externos ao repositório (neste caso, ao *MongoDB* e ao *Contentful*). Seus arquivos configuram a conexão com bancos externos e implementam métodos definidos em *core/datasources*, ao passo que fazem consultas, inserções, alterações e deleções no banco;
- **entities**: definem os modelos que devem ter os *schemas* do banco de dados, que são os objetos-entidades utilizados em todo o projeto. Aqui, no entanto, é o único lugar que vulgarmente nos distanciamos das diretrizes da arquitetura limpa de Martin, pois é também onde criamos tais *schemas*, algo totalmente atrelado à camada de *database*, mas que nos poupou um grande trabalho e organização ao custo de deixarmos nossa aplicação muito mais dependente do MongoDB;

- **presentation**: responsável por gerenciar o formato da *API* a ser consumida pelo frontend, incluindo tipos de requisição, endereços, protocolos, além de modelar também as tipagens das entradas e saídas desta interface;
- Os demais arquivos “soltos” ou dentro de **src/**, mas sem diretório são apenas arquivos de infraestrutura e configuração do projeto e ambiente de desenvolvimento.

5. Futuro

Durante o desenvolvimento, a equipe teve diversas ideias para as quais não houve tempo suficiente para aplicar. Tais ideias deveriam aprimorar a experiência de usuário e criar mais engajamento com a aplicação no futuro. Também foram tomadas decisões conscientes que apresentam baixo potencial de escalabilidade, e que poderiam ser refatoradas no futuro, caso a aplicação venha a crescer. Neste capítulo, vamos expor possíveis melhorias que podem ser implementadas no futuro.

5.1 Conteúdo

Originalmente a ideia do projeto era disseminar postagens e infográficos de maneira acessível, proporcionando uma divulgação em maior escala dos assuntos referentes à odontologia. Tendo isso em mente, existe a possibilidade de adicionar outras mídias como vídeos e podcasts. Para isso, seria necessário desenvolver uma plataforma de administração da aplicação mais robusta e flexível do que o Contentful permite.

5.2 Otimização

Devido à natureza complexa do desenvolvimento web, há diversas mudanças que poderiam ser incorporadas para a otimização de tempos de resposta e carregamento. A principal mudança, que envolveria uma alteração na infraestrutura interna da aplicação e acarretaria em um grande ganho de performance, seria o uso de uma CDN (*Content Delivery Network)* para o armazenamento do código em serviços de nuvem.

6. Experiência Pessoal

Neste capítulo, apresentaremos um relato pessoal de cada aluno que participou do projeto, contado em primeira pessoa, expondo sua jornada de aprendizado e desafios individuais no desenvolvimento. Além disso, também apresentaremos o desenvolvimento do projeto desde a escolha do objeto, bem como todos os percalços encontrados e as soluções para cada um deles. Junto com nossas experiências pessoais, buscamos explicitar todo o desenrolar da aplicação EviDent até a sua conclusão.

6.1 Luis Hikaru

6.1.1 Primeiros passos

Quando decidimos fazer uso de um projeto que já conhecíamos para a construção do Trabalho de Conclusão de Curso, eu estava bem otimista com o desenvolvimento. Não só faria uso de uma tecnologia confortável, como poderia aplicar uma Metodologia de Desenvolvimento Ágil assim como foi aprendido com o Professor Goldman no ano anterior. Interessantemente, o começo do desenvolvimento foi uma das partes que apresentaram maior quantidade de desafios. Desde a criação da arquitetura da aplicação, até a integração com o Netlify e a criação de um PWA, foram todas experiências novas e desafiadoras para mim. Demoramos mais de um mês para ter uma tela simples para apresentar à Gabriela, mas depois disso o desenvolvimento fluiu muito bem.

6.1.2 Design

A próxima dificuldade veio do fato de que nós estávamos criando, primariamente, uma interface que seria usada por pessoas em larga escala, e portanto, deveria ter boa usabilidade e uma aparência atrativa e intuitiva. Considerando que nenhum membro da equipe tem experiência prévia com conceitos de design, ainda menos com *User Interface Design,* encontramos problemas definindo a identidade visual da aplicação. Acabamos por utilizar o Figma (uma

plataforma de web design interativo e colaborativo) para definirmos juntos como seria a apresentação das nossas funcionalidades.

6.1.3 MVP

Em tecnologia, o conceito de MVP significa *Minimum Viable Product,* em outras palavras, a menor quantidade possível de software desenvolvido a fim que o produto possa ser utilizado por um usuário real. Para nós, o prazo para um MVP veio no fim de outubro, quando a Gabriela iria apresentar a aplicação em um evento que reuniria outros profissionais da área de odontologia (28º Congresso Brasileiro de Odontopediatria). Assim sendo, esse foi o período de maior intensidade do desenvolvimento, em que aumentamos a frequência de reuniões para conseguir entregar um MVP a tempo do evento.

6.1.4 Pós-MVP

Finalmente, a nossa jornada terminou com ajustes pequenos no *frontend*, e um último grande desafio: a inserção de monitoramento com o *Google Analytics*. O GA funciona através de eventos que podem ser programados, para serem disparados em momentos específicos e propriedades específicas. Dessa forma, podemos monitorar onde os usuários navegam, clicam e interagem sem termos informações diretas sobre a identidade do usuário (que seria uma violação da LGPD - Lei Geral de Proteção de Dados Pessoais).

Por sua vez, a Google oferece um console online que pode ser usado para criar uma infinidade de painéis diferentes, e possibilita a criação de “visualizações” - gráficos e tabelas customizáveis que promovem diversas interpretações diferentes dos dados colhidos. Entender não apenas quais dados gostaríamos de ter, como criar visualizações para esses dados e explicar a plataforma para a Gabriela de forma que fosse intuitiva para ambos os lados foi uma tarefa árdua, mas que gerou resultados satisfatórios.

6.2 Iggor Numata

6.2.1 O Começo

Desde que recebemos a notícia de que poderíamos dar continuidade ao projeto que preparamos no final de 2020, na disciplina de Laboratório de Métodos Ágeis, sinto que estivemos fazendo algo certo e aplicando, de fato, o que nos foi ensinado enquanto graduandos. Entregar o projeto durante a disciplina ainda cru, mas iniciado e documentado, planejado para ser desenvolvido e funcionar como aprendemos, foi, certamente, ter muita confiança no próximo time de desenvolvedores. E poder continuar a partir deste ponto, foi um alívio de que nós, do passado, preparamos o projeto da Gabriela, esperando muito e foi satisfatório ver a oportunidade de cumprir com estas expectativas.

É claro, escolhemos um caminho relativamente confortável para este projeto de conclusão, mas isso também significou que poderíamos buscar novas oportunidades de aprofundar nossas habilidades como desenvolvedores de software. O confortável tomou uma nova forma, desde as nossas primeiras reuniões, onde alinhamos as nossas expectativas para este ano trabalhando juntos e o acordado foi que a base que já possuímos, não seria o bastante para concluir o EviDent.

Foi criada uma grande promessa, de uma aplicação responsiva a diferentes telas, que entregaria uma experiência customizada aos usuários de educação e empoderamento no âmbito odontológico via *posts* e infográficos. O que, para nós, desenvolvedores, significou entregar um produto escalável, de múltiplas telas e acessível para o público. Que permita o registro de usuários com diferentes permissões, com diversos *posts* que majoritariamente carregam uma ou mais imagens. Além de um filtro de itens e agrupamento destes *posts* por categorias e subcategorias. Algo simples, mas que ainda envolve muitos conceitos diferentes e que devem conversar entre si.

6.2.2 Primeiros passos do PWA

Estagiando há 6 meses, no período em que foi iniciado o projeto, como desenvolvedor *frontend*, o começo ainda foi um pouco nebuloso se tratando de por onde deveríamos começar a desenvolver. Neste momento, confiei fortemente na experiência do meu colega Luis, enquanto trabalhava em tarefas que já tinha mais domínio. Neste começo, havia uma grande pressão em entregar algo visual e

interativo para a cliente e para os orientadores. Assim, enquanto Luis focava no que diz respeito às configurações de infraestrutura do projeto, incluindo o ambiente onde iríamos hospedar o nosso site, meu trabalho envolveu muito mais desenhar e implementar as nossas primeiras telas de autenticação. Sem nenhum tipo de integração com alguma API, propriamente dita, mas este foi também o primeiro momento em que pensamos que a aplicação deveria estar pronta para receber informações de uma API genérica que usaríamos para fazer o cadastro e login dos usuários.

6.2.3 Plataforma de administração

Após termos uma tela básica de autenticação, ainda sem muita orientação do que seria necessário, estávamos focados em entregar uma ferramenta completa principalmente para a cliente, mais do que para os usuários que consumiriam as postagens produzidas pelo EviDent. Então, pensamos que seria um bom momento para começar a imaginar os fluxos básicos que deveriam ter uma plataforma de administração para o nosso contexto: criação e gerenciamento de postagens; gerenciamento de usuários; mudanças básicas de visualização.

A partir disso, desenvolvi apenas layouts básicos para as telas de criação e edição de postagens, o que incluiu também a criação de um novo repositório, para termos as telas dos usuários e dos administradores completamente separadas. Para esta tela, foi feito um trabalho simples com formulários e que armazenavam os dados de entrada em um objeto modelado para abranger todo tipo de postagem criada até então pelo EviDent. Este objeto, no entanto, ainda não era usado, fazendo-se necessário algum banco de dados e alguma forma de populá-lo.

6.2.4 A questão do Design

Não muito por uma questão de falta de conhecimento da tecnologia, mas desde sempre avançamos muito lentamente com o nosso projeto. Muito disso deve-se à falta de alguém da área de design no projeto. O que ocasionou em algumas indefinições, por exemplo, como a tela deveria se comportar, se tamanhos, cores e, principalmente, posições dos componentes de uma tela estariam apresentáveis e agradaria o usuário final. A perspectiva do avanço do projeto caiu um pouco com este gargalo e fez-se necessário a adição de uma nova área do

desenvolvimento neste projeto ou no mínimo alguma orientação do que fazer sem que seja necessário entregar uma tela toda implementada nas reuniões com a cliente. Disto, adotamos o figma como uma maneira fácil, barata e rápida de prototipação de telas para que todos tivessem uma noção mais concreta do que estava sendo pedido e do que estava sendo entregue.

Concordamos então em trabalhar sobre *layouts* temporários desenhados por nós, *devs*, e pela cliente, algo ainda cru, mas que já nos dava muito mais segurança. O que nos levou ao grande destaque da aplicação: a tela de *posts*. A partir de um design simples que concebemos, ficou claro quais informações deveríamos ter em um post, o que poderia ser confuso para o usuário e o quanto seria difícil importar os infográficos, o coração das postagens feitas pela entidade por trás do EviDent, de um formato criado para o Facebook e Instagram para o nosso *app* responsivo.

6.2.5 Post.json - O nosso primeiro banco de dados

Com o aval para construir a tela de *posts*, ficou claro que precisaríamos importar os mesmos dados das postagens que já existiam. Assim, a necessidade de um servidor *backend*, que gerenciasse os dados armazenados em um sistema gerenciador de banco de dados, tornou-se uma discussão muito mais relevante já que pretendíamos entregar um produto escalável. Isso, no entanto, representava uma mudança de escopo bem grande no nosso roteiro previsto para o projeto, pois nenhum de nós conhecia, de fato, como gerir um servidor *backend* até então.

Tendo em vista a situação, fomos orientados a não nos preocupar com isso, pois para um primeiro momento, as entregas visuais eram muito mais importantes e traziam muito mais valor para a cliente. Então, seguimos uma estratégia menos escalável, de difícil manutenção e manipulação: guardar os dados dos *posts* no nosso próprio código. Isto, claro, não resolveria o nosso problema com a autenticação à nossa aplicação, mas seria um paliativo mais concreto do que a cliente poderia esperar das telas e de forma interativa.

O que entregamos era não mais que apenas um protótipo de alto nível que consumia dados de um arquivo JSON (JavaScript Object Notation), um formato de

arquivo muito utilizado para armazenar, ler e transmitir dados, muito útil dado que estamos trabalhando em um *framework* construído em JavaScript. E para os nossos fins de validação e modelagem de dados, bem como apresentar visualmente esta primeira grande entrega, serviu bem ao seu propósito. A cliente ficou bem satisfeita e conseguiu realizar alguns testes com usuário dentro do círculo mais interno ao EviDent. De fato, foi uma etapa muito importante e muito frutífera para o nosso desenvolvimento, pois a partir destes testes, obtivemos *feedbacks* que pudemos converter em uma interface mais refinada posteriormente.

Ainda assim, essa estratégia se provaria ser extremamente limitada, ao passo que para cada *post* novo, deveria ser alterado o arquivo dentro do nosso repositório do projeto e então deveria ser feito um novo lançamento de versão do site. Obrigando a cliente a sempre ter algum desenvolvedor em sua equipe. A partir deste ponto, nosso desenvolvimento no que diz respeito ao "*frontend*" e apesar do repositório representar todo o projeto até o momento, foi muito mais focado em refinamentos das telas que já havíamos apresentado e algumas outras configurações que trariam muito mais a cara de um *App* nativo para o nosso PWA.

6.2.6 Monolito vs. *Frontend* & *Backend*

Dado o escopo do nosso projeto, não seria estranho continuar com o seu desenvolvimento inteiramente em apenas um repositório. Afinal, não era uma aplicação com muita complexidade, que apresentaria muitas interações com os usuários e que seria atualizada com uma baixa frequência, tornando mudanças e manutenção mais viáveis apesar das diversas responsabilidades desde mostrar a camada de visualização até atualizar o banco de dados.

Porém, nosso objetivo era fazer algo melhor do que o EviDent já fora um dia. Que aplicasse boas práticas de desenvolvimento e que comportasse tanto crescimento quanto ambicionava a cliente. Assim, aplicando o que aprendemos em nossos estágios, concluímos com os orientadores que a abordagem correta seria dividir o projeto em duas frentes de desenvolvimento: *frontend* - responsável por mostrar os dados da forma correta, interagir com o usuário, prezar pela experiência do usuário como um todo; e *backend* - responsável por interagir com o sistema gerenciador de banco de dados, fazer o tratamento, manipulação e validação dos

dados, e fornecer os dados de forma eficaz e organizada para o *frontend* na forma de uma interface (API, Interface de Programação de Aplicativos)

Em conclusão, a ideia de separar as frentes progrediu como esperávamos e a cliente compreendeu o novo ritmo de desenvolvimento. Afinal, não seria interessante parar completamente com o desenvolvimento da aplicação no *frontend* e nenhum de nós tinha uma boa experiência em desenvolvimento de um servidor *backend*. Para esta tarefa, assumi a frente de *backend*, por ser um objetivo de interesse pessoal e profissional, e seria uma ótima oportunidade para realmente estudar e aplicar os novos conhecimentos e boas práticas.

6.2.7 *Backend* e Login, um *discovery* técnico

Antes de começarmos a descrever esta parte do projeto, é importante explicar brevemente o que é o *discovery* técnico, porque é fundamental nesta etapa do desenvolvimento da aplicação e como eu o conduziria. *Discoveries* em geral são pesquisas, estudos e testes que possuem a intenção de determinar a viabilidade de alguma solução. Neste contexto, seria o estudo da possibilidade de implementar um *backend* e quais seriam as melhores ferramentas que o comporiam. É importante que sejam avaliados os benefícios e riscos de cada ponto desta implementação, incluindo a estimativa de tempo e esforço que demandaria esta tarefa. Esta investigação poderia vir acompanhada de uma prova de conceito (PoC), que neste caso pode-se ser entendido como uma funcionalidade correta e completa, que para o primeiro passo do servidor seria a autenticação do usuário.

Nossa perspectiva neste ponto do projeto era de, no máximo, mais três meses de desenvolvimento até a entrega de um produto mínimo viável (MVP) e mais dois meses, divididos entre ajustes pequenos e a redação desta monografia. O tempo era confortável, mas com pouca margem para erros nessas estimativas. Ainda assim, tornou-se imprescindível que fosse feito um *backend* para abrir caminho para as funcionalidades que mais desejava a cliente: uma experiência personalizada para cada usuário, e sem login na aplicação, esta demanda seria impossível de ser alcançada.

O que planejamos em conjunto com João Francisco Daniel, um dos orientandos do professor Goldman, que nos auxiliou desde o início, e tem grande

experiência como desenvolvedor web, foi a criação de um *backend* com ferramentas simples ou familiares para nós e que utilizasse um banco de dados que atendesse bem às nossas demandas, sem apresentar uma grande curva de aprendizado. Diminuindo ao máximo o atrito com as novas tecnologias, o *discovery* poderia ser concluído a tempo de criarmos as integrações entre *backend* e *frontend* a tempo da prevista entrega.

6.2.8 *Backend* - Primeiras versões

Com confiança na orientação do orientando João, pude prosseguir com meus estudos sobre como implementar o nosso *backend*. Houve algumas interações deste servidor, o que em parte é documentado pelo git do projeto, mas ainda não reflete o processo em sua totalidade. A primeira versão do servidor foi uma cópia de um *template*, modelo base, do que parecia ser exatamente o que procurávamos: um servidor em Node.JS, Express, MongoDB. Pesquisando um pouco, é possível ver que essa é uma web *stack*, conjunto de ferramentas de software usados para desenvolver a aplicação, bem comum no mercado de desenvolvimento web.

O *template* tinha muitas vantagens, dentre elas, trazer uma arquitetura completa, com diversos exemplos já implementados a incluir todo um fluxo completo de login e alguns testes já implementados. O único porém: todo o código foi escrito em JavaScript, algo que teria grande potencial para dificultar o nosso projeto caso tentássemos substituir a linguagem ou tornaria o desenvolvimento muito mais complicado. Além disso, a arquitetura era diferente da qual estávamos acostumados a trabalhar, então ainda haveria essa curva de aprendizado.

Por estes motivos, concluímos que esse *template* tinha potencial, mas não o suficiente para se encaixar no nosso contexto. Tivemos que recomeçar e procuramos algo mais próximo do que estávamos confortáveis. A segunda versão apresentava muitas boas mudanças, as tecnologias eram exatamente as que precisávamos com mais algumas bibliotecas que garantiam um código até mesmo mais saudável do que imaginávamos, mas infelizmente, a arquitetura, mais uma vez causou um pouco de atrito para me acostumar a como implementar novas funcionalidades.

6.2.9 *Backend* - Um recomeço

A estratégia de iniciar o desenvolvimento em cima da estrutura já pronta por algum terceiro não estava dando frutos e o tempo disponível estava acabando. Foi necessário repensar como deveria ser este começo, focando agora em aproveitar os pontos fortes dos projetos que já tivemos contato e adaptá-los para o nosso contexto de um projeto simples o suficiente para permitir novos desenvolvedores começarem e robusta o suficiente para comportar mudanças não previstas. Chegou a hora de perder um pouco a pressa e apostar muito mais no estudo do que já conhecíamos para criar algo satisfatório. Disso, estudei as tecnologias presentes nas primeiras versões e entendi porque elas faziam sentido para a nossa *stack* e como poderíamos trazer elas para uma proposta de arquitetura que queríamos seguir.

O ponto de partida, então, tornou-se os projetos com os quais tivemos contato durante nosso período de estágio que foram o nosso ponto inicial também para o *frontend*, então era algo esperado de acontecer, mas sem copiar nenhum arquivo, por motivos legais. Com esta decisão, pudemos adotar a proposta da **Arquitetura Limpa**, um design de arquitetura de projetos proposto por **Robert Cecil Martin**, popularmente conhecido como ***Uncle Bob***, em seu livro e blog *Código Limpo*. Essa arquitetura, então, ditou como seria feita toda implementação desta nova versão.

Foi criado um novo projeto, limpo, apenas com os arquivos essenciais. Coloquei algumas bibliotecas interessantes, unindo experiências recentes e de todo o aprendizado adquirido na disciplina de metodologia ágil e no estágio. E procurei extensivamente por guias e tutoriais na internet que pudessem guiar esta etapa de configuração da infraestrutura do projeto. No final de duas semanas, tivemos um servidor simples ainda sem nenhuma funcionalidade, mas que poderia finalmente ser a base para o nosso desenvolvimento.

6.2.10 Login, a primeira funcionalidade

Enfim, era a hora de implementarmos um login de usuários. Além de tudo, o estudo intenso feito para a etapa de configuração seria dobrado neste momento para aprender a implementar, de fato, novas funcionalidades. Isso envolveu, primeiramente, aprender a criar o servidor, com o Express.js, aprender a utilizar o MongoDB e o Mongoose e aprender a utilizar uma biblioteca fundamental para o uso da nossa arquitetura, o **typedi**. Essa biblioteca é a que possibilita o conceito de

“injeção de dependência” entre as camadas, que envolve utilizar implementações de classes de diferentes camadas da aplicação, fazendo com que as camadas não precisem conhecer como acontece tal implementação.

Visando adquirir estes conhecimentos, busquei por mais alguns tutoriais, agora visando como implementar, tornar o “login” funcional e me apoiei em repositórios do meu estágio para aplicar os novos conhecimentos com boas práticas. O material base principal para esta etapa foram alguns vídeos do canal **codedamn** no YouTube, que são apresentadas as tecnologias em maior profundidade e como utilizá-las na nossa aplicação. Em adição, foi necessário aprender a utilizar algumas novas tecnologias e bibliotecas para melhor lidar com alguns detalhes do servidor: **routing-controllers**, para facilitar o uso do express e organizar com mais facilidade os endereços da API; **typegoose**, para implementar os modelos a serem usados pelo mongoose de forma mais amigável para o TypeScript; **bcrypt** e **jwt**, para lidar com autenticação e validação de usuários via *tokens*; e **class-validator**, para lidar com a validação de dados que são recebidos e enviados para o servidor. A maioria dessas bibliotecas não são essenciais para o andamento do servidor, mas definitivamente trazem muita facilidade para o desenvolvedor.

Assim, com mais duas semanas, foi criada a funcionalidade completa do fluxo de cadastro e login. Bastava agora disponibilizar a nossa API para ser consumida pelo *frontend*, pois até então, estávamos fazendo testes locais do produto, utilizando as ferramentas auxiliares de desenvolvimento **Postman**, que nos permite fazer requisições para API’s e obter respostas em um formato inteligível para pessoas e **Robo3T**, uma versão local do MongoDB.

6.2.11 Implantação no Heroku

Restava, então, implantar o nosso servidor em alguma plataforma. Como já havíamos planejado utilizar o Heroku pela facilidade e pelo suporte que oferece para projetos em Node.js. Houve, no entanto, alguns aprendizados necessários. De início, tive que aprender como utilizar a interface de linha de comando do Heroku para facilitar ao máximo futuras documentações sobre como realizar a implantação de novas versões. Nada complexo demais, mas foi necessário desde então como configurar o ambiente da plataforma para comportar diferentes versões do projeto.

Gostamos de seguir boas práticas de versionamento semântico e diferentes ambientes de uso da aplicação. Por esses motivos, primeiro foi necessário configurar cada ambiente que teríamos: ***development*** para testes menores, uso e desenvolvimento local; ***staging*** para testes da cliente, orientadores e alguns testadores de versões beta da cliente; e ***production*** para o usuário final. O que incluía configurar corretamente variáveis de ambiente, como chaves de acesso e endereços para os diferentes bancos de dados, e preparar comandos simplificados para os três ambientes diferentes, causando assim o mínimo de atrito possível com os futuros desenvolvedores.

Com tudo preparado, foi apenas uma questão de integrar a nova API no projeto do *frontend*. A partir de agora, a maior parte do trabalho seria replicar de alguma forma o que foi feito para a implementação do login, variando apenas em como seriam feitas as consultas ao banco de dados e como lidar

íamos com os modelos das nossas novas entidades. A mais notória delas, novamente, a entidade **Post**.

6.2.12 Posts em um novo formato e o problema das imagens

Os modelos dos *posts* da nossa aplicação foram provavelmente o modelo de dados que mais evoluiu desde da primeira prototipação o comportamento completo de visualizar o conteúdo que já existia nas redes sociais do EviDent, agora no nosso produto. Foram feitas algumas alterações desde que criamos o “Post.json”, para nos adaptarmos às necessidades que surgiam, mas traduzir tudo para o nosso banco de dados implicava muitas mudanças. Já não deveria ser necessário que a cliente tivesse constantemente uma equipe de desenvolvedores a acompanhando para fazer atualizações com regularidade no seu conteúdo. Para isso, deveríamos pensar em como desvincular a criação de novas postagens do código desenvolvido, o que não era possível até então por sempre necessitarmos fazer o *upload* das imagens, o núcleo das postagens produzidas pela entidade.

Normalmente, nós pensamos em utilizar uma **Rede de Fornecimento de Conteúdo** (CDN) para resolver este problema. Uma CDN seria mais uma camada de servidores na nossa arquitetura completa, responsável por armazenar e distribuir os arquivos de imagens por meio de URL’s de forma fácil e performática. O problema

era como fazer e em menos de um mês. Estudamos algumas alternativas de CDN, mas apenas uma parecia viável dentro do orçamento para a tarefa, que sempre deveria ser sem custo durante o período de desenvolvimento. Chegamos, assim, em um serviço adicional do Heroku **Cloudinary**, dedicado especialmente à hospedagem de imagens e vídeos e de fácil integração com aplicações implantadas no Heroku. Seria um pouco trabalhoso, aumentaria bastante a complexidade do que fazíamos no momento, mas ao longo prazo seria exatamente o que precisaríamos.

No fim, decidimos não seguir pela abordagem do Cloudinary, a pressão da entrega era muito maior do que poderíamos lidar no momento, enquanto estudávamos toda uma nova tecnologia para adicionar no servidor. Assim, o João sugeriu uma abordagem paliativa muito simples: hospedar as imagens todas no Google Drive da entidade e apenas utilizar as URL's do compartilhamento aberto. De fato, era a nossa melhor aposta e nos permitiria começar a trabalhar em ajustes finos para fazer uma boa entrega do produto mínimo viável (MVP).

6.2.13 Pós-MVP e Contentful

Finalizamos a entrega, momento em que o produto seria compartilhado para o grande público alvo do EviDent e seriam investidos os esforços da equipe de marketing da entidade. Agora, poderíamos focar em apenas melhorar a experiência do nosso segundo usuário, o administrador da aplicação. O último item do nosso escopo de tarefas era uma ferramenta eficiente e segura para permitir que a cliente tivesse liberdade de facilmente gerenciar as postagens, além de podermos começar a focar nossos esforços nesta monografia. Além disso, era o período de entregas do meio do semestre. Nosso foco ficava bem dividido entre momentos diferentes de um único dia. Contudo, o mais importante ainda era pensar em alguma estratégia de fazer essa plataforma de administrador acontecer.

A primeira abordagem foi criar uma nova permissão para os usuários do nosso site. Com essa permissão, poderíamos fazer mudanças ao acessar páginas ocultas para o público geral. Havia apenas um porém: fazer funções e permissões diferentes para usuários mostrou-se ser uma tarefa de grande complexidade e com vários desafios de segurança. Logo fomos dissuadidos de seguir com o plano. Com este novo bloqueio, uma última vez, o orientando João conseguiu trazer mais uma

visão experiente para o nosso problema: usar um sistema de gerenciamento de conteúdo (CMS), recomendando o **Contentful**, cuja proposta estudamos os benefícios e dificuldades, mas uma vez que fomos bem orientados, mostrou-se ser algo bem simples.

Com o Contentful, agora seria possível criar postagens dentro do formato desejado pela cliente, em uma plataforma intuitiva e com grande capacidade para escalabilidade, enquanto não tivéssemos nossa própria plataforma. Bastou configurar o formato desejado para inserção de novas postagens de acordo com o modelo que havíamos preparado para o nosso banco de dados e adicionar o Contentful como um outro banco de dados para a nossa aplicação, de onde apenas faríamos consultas.

A estratégia foi a parte mais complicada desta implementação. Foi necessário lidar com algumas trocas de performance, pois não seria muito viável comparar toda postagem já no nosso banco de dados principal com o conteúdo atualizado pelo Contentful e também não seria uma boa prática incluir uma requisição para o Contentful para toda requisição que fizéssemos da aplicação para o nosso servidor backend. O meio termo a que chegamos foi: se faz mais de uma semana que o nosso banco principal não é atualizado, apagamos todos os dados desta coleção de Posts, fazemos uma requisição de todas as postagens do Contentful e repopulamos o MongoDB. Caso contrário, em menos de uma semana que ocorreu essa atualização, sempre puxamos os dados do MongoDB.

A ideia é que essa estratégia dê autonomia o suficiente para a equipe de criação de conteúdo do EviDent gerenciar as postagens como bem entenderem até que seja implementada uma plataforma interna para este mesmo trabalho, mas agora com a liberdade de atualizar os posts com maior dinamicidade.

7. Conclusão

Após apresentarmos o projeto EviDent, seu objetivo e meta, além de todas as ferramentas e metodologias utilizadas neste projeto, bem como todo o desenvolvimento da aplicação, faz-se necessário um capítulo para apresentarmos nossas considerações finais enquanto desenvolvedores da plataforma, mas também trouxemos a visão final da nossa cliente, Gabriela Machado, para concluirmos nossa jornada como a Evident.

7.1 Iggor Numata

Quando aceitamos reentrar nesta aventura em janeiro deste ano, não conseguiria prever ou acreditar no quanto eu cresci durante este projeto. Não é algo novo trabalhar com o desenvolvimento de uma aplicação, mas ainda assim, foi uma experiência única. Parte da liberdade de um projeto pessoal, mas toda a responsabilidade de um projeto com uma cliente, desta vez, sem a costumeira ajuda de uma equipe ou líder técnico para nos guiar. A primeira impressão que levarei deste trabalho de conclusão é, com certeza, o quanto meu tempo de graduação me preparou para um desafio como este, e o quanto este tempo ganhou em me equipar para o futuro. Felizmente, esta foi a maior oportunidade dentro do contexto universitário que tive para exercer fundamentos para um desenvolvedor de software. A mais importante de todas: saber como lidar com problemas.

Então, vem o olhar do meu lado profissional, que enquanto estudava em um estágio de desenvolvimento de software e trazia para o EviDent todo o bom conhecimento que adquiri neste tempo, encarava tudo isso como uma provação e um ambiente para aplicar e aprofundar novas práticas. Enxergo claramente como o meu lado profissional e o lado acadêmico se beneficiaram durante o desenvolvimento da nossa aplicação, uma troca mútua, que resultou na realização pessoal de saber de verdade o que estou fazendo e porquê é algo bom.

Por fim, tive a oportunidade de evoluir como pessoa e um ser social. Foi uma experiência melhorar minha comunicação através de conversas com a cliente e com os professores. E principalmente, aprender a lidar com o fato de estar trabalhando,

estudando em uma graduação, fazendo um TCC e sendo amigo de alguém que confiou tanto em mim, tanto quanto eu sempre confiei nele. O convívio que adquiri com o Hikaru e com o resto do nosso time é algo que levarei como aprendizado imensurável para a vida.

7.2 Luis Hikaru

Para mim, a jornada de construção do projeto apresentou um aprendizado muito maior em questão de organização de software do que da programação em si. Atualmente, com as demandas por aplicações complexas e extensas aumentando, a questão de **qual é a melhor forma de desenvolver software em equipe** torna-se um tópico cada vez mais relevante para a sociedade como um todo e explorar os conceitos de desenvolvimento ágil, integrando sua filosofia no meu dia-a-dia, trouxe-me um entendimento mais profundo do que é qualidade de código.

Meu maior desafio durante o desenvolvimento do projeto foi adquirir a noção de que o software que eu estava criando deveria ser utilizado por pessoas em larga escala, e, por isso, manter sempre em mente a experiência de usuário ao criar novas telas, menus e botões. Com isso, foi mais fácil mapear casos de uso e resultou em um software melhor.

São pouquíssimas as situações no período de graduação em que os alunos são ensinados conceitos práticos de desenvolvimento em equipe, escalabilidade de software e boas práticas de manutenção e documentação de código, e esses foram os maiores aprendizados que tive na jornada do EviDent. Além disso, a experiência de poder fazer o TCC em equipe foi fantástica. Saber que a Gabriela e o Iggor dependiam dos meus esforços adicionou um elemento a mais de seriedade e comprometimento que elevou meus esforços, e saber que poderia contar com o Iggor em períodos difíceis foi um grande alívio. Finalmente, gostaria de estender meus agradecimentos ao professor Goldman e ao orientando João que nos mentoraram extensivamente durante o projeto, adicionando sugestões e conselhos importantíssimos.

7.3 Gabriela Machado

Participar do processo de TCC do Iggor e do Luis foi uma experiência muito agradável como cliente. Já os conhecia (virtualmente) devido a disciplina MAC0472 - Laboratório de Métodos Ágeis e fiquei feliz pelo projeto do EviDent ser escolhido por eles anteriormente, e agora toparem aprimorá-lo como trabalho de conclusão de curso. Foi possível acompanhar o desenvolvimento dos alunos durante o processo e perceber a aplicação do conhecimento adquirido previamente na graduação, em que eles traziam ideias baseados nos conhecimentos das disciplinas, e de novos conhecimentos adquiridos durante a elaboração e desenvolvimento do TCC. Como cliente (e dentista) temos uma ideia do que queremos e os alunos, como profissionais, conseguiram me mostrar o caminho que estávamos percorrendo e ate onde seria possível ir.

O Iggor é um aluno muito disciplinado e responsável, sempre se esforçou para realizar as demandas que eu trazia às reuniões, e quando não era possível se esforçava para encontrar uma alternativa. Luis é um aluno com uma visão profissional e prática, foi um aluno resolutivo e me ajudava a entender até onde seria possível ir com a plataforma. Foi perceptível e importante como cliente me sentir ouvida e ver o empenho em entender as demandas para propor recursos que fossem relevantes para a iniciativa. Agradeço a preocupação constante de transpor o técnico para minha linguagem. Ambos foram extremamente amigáveis e agradeço as conversas e a relação que criamos!

Sou imensamente grata pelo empenho de vocês no projeto, sempre tentando fazer o excelente não só o que era proposto como trabalho de conclusão de curso. Gostaria de aproveitar para agradecer ao João que também foi parte fundamental no desenvolvimento. E por último e com certeza não menos importante gostaria de agradecer ao Prof. Alfredo que tenho um carinho imenso, obrigada por mais uma oportunidade de desenvolver nossa ideia! Como aluna de pós graduação cresci muito durante todo esse processo, além da parte odontológica na FOUSP vocês me deram a oportunidade de sair da caixinha e aprender a trabalhar com plataformas diferentes, além de desenvolver a habilidade (ainda em construção) de vender minha ideia, pensando em patrocinadores e meus clientes finais.

Iggor e Luis, vocês serão ótimos profissionais e tenho certeza que realizarão muitos sonhos. Não esqueçam que tudo é um processo, uma maratona, que agora

vocês estão dando a largada... não adianta ter pressa, um dia de cada vez e um aprendizado por vez! Estarei sempre torcendo, obrigada (:

8. Referências

Livros referenciados para o desenvolvimento do projeto:

- Martin, Robin (2008) - *Clean Code: A Handle of Agile Software Craftmanship*
- Martin, Robin (2017) - *Clean Architecture: A Craftsman's Guide to Software Structure and Design*

Documentações e guias das tecnologias usadas ou citadas:

- Atlas MongoDB - https://www.mongodb.com/atlas/database
- Axios - https://axios-http.com/docs/intro
- bcrypt.js - https://github.com/dcodeIO/bcrypt.js#readme
- class-validator - https://github.com/typestack/class-validator
- Cloudinary - https://elements.heroku.com/addons/cloudinary
- Contentful - https://www.contentful.com/developers/docs/
- cors - https://github.com/expressjs/cors#readme
- Day.js - https://day.js.org/docs/en/installation/installation
- Google Analytics - https://developers.google.com/analytics
- Heroku - https://devcenter.heroku.com/categories/reference
- Manifesto Ágil - https://agilemanifesto.org/
- Material UI - https://v4.mui.com/getting-started/usage/
- MongoDB - https://docs.mongodb.com/manual/
- Mongoose - https://mongoosejs.com/docs/guide.html
- Node.js - https://nodejs.org/en/docs/
- Netlify - https://docs.netlify.com/?_ga=2.252538377.1202787913.1640390756-215534542.1630604561
- React.JS - https://reactjs.org/docs/getting-started.html
- routing-controllers - https://github.com/typestack/routing-controllers
- TypeDI - https://docs.typestack.community/typedi/
- Typegoose - https://typegoose.github.io/typegoose/docs/guides/
- TypeScript - https://www.typescriptlang.org/docs/
- uuid - https://github.com/uuidjs/uuid#readme

Artigos e tutoriais utilizados:

- Adelakun, Ayobami (2020) - *Typescript With MongoDB and Node/Express* - https://medium.com/swlh/typescript-with-mongoose-and-node-express-24073d51d2ee
- Castro, Daniel (2020) - *Configurando Node.js com TypeScript, nodemon e Jest* - https://danieldcs.com/configurando-node-js-com-typescript-nodemon-e-jest/
- codedamn (2020) - *MongoDB + Mongoose + Node.js Crash Course | CRUD and fundamentals of MongoDB in* YouTube - https://youtu.be/b91XgdyX-SM
- codedamn (2020) - *Node.js Register & Login Tutorial - Learn how to authenticate with Node.js, MongoDB and JWT in* YouTube - https://youtu.be/b91XgdyX-SM
- Daniel Corcoran (2020) - *Integrating Contentful with React & Typescript* - https://danielcorcoranssql.wordpress.com/2020/08/26/integrating-contentful-with-react-typescript/
- “Data Pilot” (2019) - *Simple mongoose and node js Example* - https://kb.objectrocket.com/mongo-db/simple-mongoose-and-node-js-example-1007
- Harms, Joshua (2021) - *Implementing a JWT auth system with TypeScript and Node* - https://nozzlegear.com/blog/implementing-a-jwt-auth-system-with-typescript-and-node
- Heinzmann, Emilio (2019) - *ESLint + Prettier, a dupla perfeita para produtividade e padronização de código.* - https://medium.com/cwi-software/eslint-prettier-a-dupla-perfeita-para-produtividade-e-padronização-de-código-6a7730cfa358
- Jason Watmore (2020) - *Node.js - Role Based Authorization Tutorial with Example API* - https://jasonwatmore.com/post/2018/11/28/nodejs-role-based-authorization-tutorial-with-example-api
- Karpov, Valeri (2019) - *How find() Works in Mongoose* - https://thecodebarbarian.com/how-find-works-in-mongoose.html

- “MDN Contributors” (2021) - *Tutorial Express Parte 3: Usando um banco de dados (com Mongoose)* - https://developer.mozilla.org/pt-BR/docs/Learn/Server-side/Express_Nodejs/mongoose
- Sodeeq, Elusoji (2018) - *How to deploy from one local app to multiple heroku apps.* - https://medium.com/@sdkcodes/how-to-deploy-from-one-local-app-to-multiple-heroku-apps-3fe02a5fcf64
- The Net Ninja (2021) - *Next.js & Contentful Tutorial in* YoutTube - https://www.youtube.com/playlist?list=PL4cUxeGkcC9jClk8wl1yJcN3Zlrr8YSA1
- Türkmenoğlu, Mert (2021) - *Dependency Injection in TypeScript.* - https://levelup.gitconnected.com/dependency-injection-in-typescript-2f66912d143c
- How to convert a Title to a URL slug in jQuery? *in* StackOverflow - https://stackoverflow.com/questions/1053902/how-to-convert-a-title-to-a-url-slug-in-jquery
- Remove accents/diacritics in a string in JavaScript *in* StackOverflow - https://stackoverflow.com/questions/990904/remove-accents-diacritics-in-a-string-in-javascript
- REST Resource Naming Guide - https://restfulapi.net/resource-naming/