UNIVERSIDADE FEDERAL DO CEARÁ

INSTITUTO UNIVERSIDADE VIRTUAL

CURSO DE GRADUAÇÃO EM SISTEMAS E MÍDIAS DIGITAIS

GABRIEL DE MELO BASTOS

O PROCESSO DE DESENVOLVIMENTO E DESIGN DA APLICAÇÃO WEB OFERTA FÁCIL PARA ALOCAÇÃO DE TURMAS DO CURSO DE SISTEMAS E MÍDIAS DIGITAIS

FORTALEZA

2022

GABRIEL DE MELO BASTOS

O PROCESSO DE DESENVOLVIMENTO E DESIGN DA APLICAÇÃO WEB OFERTA FÁCIL PARA ALOCAÇÃO DE TURMAS DO CURSO DE SISTEMAS E MÍDIAS DIGITAIS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Sistemas e Mídias Digitais do da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Sistemas e Mídias Digitais.

Orientador: Prof. Dr. Rafael Augusto Ferreira do Carmo

FORTALEZA

2022

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

B328p Bastos, Gabriel de Melo.
O processo de desenvolvimento e design da aplicação web oferta fácil para alocação de turmas do curso de sistemas e mídias digitais / Gabriel de Melo Bastos. – 2022.
51 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Instituto UFC Virtual, Curso de Sistemas e Mídias Digitais, Fortaleza, 2022.
Orientação: Prof. Dr. Rafael Augusto Ferreira do Carmo.

1. Angular. 2. Timetabling. 3. Oferta de turmas. 4. Open source. I. Título.

CDD 302.23

GABRIEL DE MELO BASTOS

O PROCESSO DE DESENVOLVIMENTO E DESIGN DA APLICAÇÃO WEB OFERTA FÁCIL PARA ALOCAÇÃO DE TURMAS DO CURSO DE SISTEMAS E MÍDIAS DIGITAIS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Sistemas e Mídias Digitais do da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Sistemas e Mídias Digitais.

Aprovada em:

BANCA EXAMINADORA

Prof. Dr. Rafael Augusto Ferreira do Carmo (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. Antônio José Melo Leite Júnior
Universidade Federal do Ceará (UFC)

Prof. Dr. Matheus Henrique Esteves Paixão
Universidade Estadual do Ceará (UECE)

Dedico esse trabalho a minha mãe por acreditar em mim, no meu potencial e em minha perseverança.

AGRADECIMENTOS

A minha mãe por desde cedo, ter me apoiado durante meus estudos e em todas as dificuldades que surgiram ao longo dos anos.

Ao Prof. Dr. Rafael Augusto Ferreira do Carmo, pela orientação e pelo auxílio no desenvolvimento deste trabalho e ter aceito me orientar no momento que eu mais precisei.

Aos amigos e colegas no curso Sistemas e Mídias Digitais que me apoiaram nos momentos mais difíceis.

A minha amiga Jordana que me ajudou a me manter são quando parecia que nada ia dar certo.

RESUMO

A alocação das ofertas acadêmicas não é um processo simples ou rápido. A cada semestre as coordenações de curso e chefias de departamento se deparam com a necessidade fazer a alocação das salas e professores para suprir a necessidade de turmas dos diferentes cursos existentes, respeitando as diferentes restrições e as demandas do corpo discente. A Resolução do Problema de Timetabling através de métodos manuais é uma prática comum. Esse método apesar de resultar em soluções que cumprem praticamente todas as restrições demanda muito tempo e recursos, além de que em escalas maiores não garante um bom resultado. Nessa condição, este trabalho tem como objetivo descrever e documentar todo o processo de desenvolvimento da aplicação web *open source* "Oferta Fácil"desde os métodos que levaram a sua ideação até a produção de seu produto viável mínimo(MVP), destacando o processo de gerenciamento do projeto. A aplicação tem como objetivo aliviar os pontos negativos dos métodos manuais reduzindo processos repetitivos e trabalhosos, além de melhorar a visualização de informações importantes no curso de Sistemas e Mídias Digitais.

Palavras-chave: Angular. Timetabling. Oferta de turmas. Open source.

ABSTRACT

The allocation of academic offerings is not a simple or quick process. Each semester, the course coordinators and department heads are faced with the need to allocate rooms and professors to meet the need for classes of the different existing courses, respecting the different restrictions and demands of the student body. Solving the Timetabling Problem using manual methods is a common practice. This method, despite resulting in solutions that meet virtually all constraints, requires a lot of time and resources, and on larger scales it does not guarantee a good result. In this condition, this work aims to describe and document the entire development process of the open source web application "Oferta Fácil"from the methods that led to its ideation to the production of its minimum viable product (MVP), highlighting the management process from the project. The application aims to alleviate the negative points of manual methods by reducing repetitive and laborious processes, in addition to improving the visualization of important information in the Digital Systems and Media course.

Keywords: Angular. Timetabling. Classes offer. Open source.

SUMÁRIO

1 INTRODUÇÃO

O processo de alocação de professores, salas e laboratórios na formação de turmas das diversas disciplinas de um curso é um processo repetitivo. Tendo como exemplo uma instituição de ensino superior, a cada semestre ou ano letivo, as coordenações de curso e chefias de departamento se deparam com a necessidade fazer a alocação das salas e professores para suprir a necessidade de turmas dos diferentes cursos existentes. Esta tarefa é repetida regularmente pois as demandas por turmas são variáveis, dependendo da quantidade de alunos aprovados ou reprovados, interesse específico do corpo discente em determinada disciplina ou pela pequena mas constante mudança do corpo docente de um determinado curso, quando um docente se afasta por problemas de saúde ou por outros motivos, por exemplo.

Além disso, tanto salas e laboratórios, bem como professores, são recursos escassos e deve-se fazer uma boa utilização dos mesmos, tendo em vista um bom e adequado uso dos espaços físicos disponíveis e o respeito à carga de trabalho dos docentes de uma dada instituição. Desta forma, este trabalho de alocação otimizada de recursos físicos e recursos humanos é naturalmente complexo e deve ser estudado em profundidade.

Tal problema pode ser resolvido de forma computacional ou de forma manual. Ao se resolver este problema de forma computacional, como visto neste próprio documento, faz-se uso de diversos algoritmos de otimização combinatória que recebem como entrada diversas características dos recursos disponíveis e efetuam um cálculo para se chegar a uma solução de alocação ótima destes diversos segundo um critério específico selecionado previamente. Se resolvido de forma manual, um especialista, normalmente o coordenador de curso ou diretor de unidade acadêmica, faz a alocação dos diferentes recursos seguindo uma heurística própria e seu conhecimento específico das diferentes nuances de salas, laboratórios e professores disponíveis, tendo como critério de otimalidade sua própria percepção da qualidade da alocação proposta.

Um método garante uma solução ótimo para um dado critério mas pode não ter como entrada características importantes dos recursos para o processo de alocação. Já o outro método depende de um especialista e a qualidade da solução apresentada depende da avaliação subjetiva deste especialista, ou seja, ambos os métodos têm seus lados positivos e negativos. Conforme discutido em Kristiansen e Stidsen (2013), a literatura indica que soluções computacionais são ideias para grandes instâncias deste problema, contendo muitas dezenas e até centenas ou milhares de salas, laboratórios, professores, disciplinas e turmas em que se pode ignorar muito das subjetividades do processo. Já as soluções manuais, podem ser utilizadas em pequenas

instâncias do problema, onde o conhecimento de causa do especialista é suficiente para construir soluções boas o suficiente, levando em consideração as muitas características e restrições de todos os recursos disponíveis. O foco deste trabalho é o estudo e melhoria do processo de alocação para uma instância do problema de pequeno porte no curso de Bacharelado em Sistemas e Mídias Digitais (SMD) da Universidade Federal do Ceará (UFC).

Para tal, é estudado o processo de alocação de salas, laboratórios e professores ocorrido no curso SMD. Neste contexto, o processo de alocação ocorre semestralmente, onde aproximadamente 40 professores, 5 salas, 1 ateliê e 6 laboratórios são base para o funcionamento de pouco menos 90 turmas, divididas nos turnos matutino, vespertino e noturno. Atualmente, esta atividade é efetuada de forma manual pelo conjunto de coordenadores do SMD Diurno e SMD Noturno com auxílio dos colegiados dos cursos, visto que o sistema de gerenciamento acadêmico da universidade, o Sistema Integrado de Gestão de Atividades Acadêmicas (SIGAA), não possui um módulo que execute tal função de forma automatizada, fazendo com que estes atores utilizem ferramentas e técnicas próprias, muitas vezes inadequadas, para executar esta tarefa a cada semestre letivo.

Dado o exposto, este trabalho tem como **objetivo geral** descrever o processo de desenvolvimento de um MVP de uma solução para o problema de alocação de salas, laboratórios e professores ocorrido no SMD. Para alcançar essa meta, temos como **objetivos específicos**:

- Apresentar como a literatura discute as muitas versões do problema de alocação
- Listar alternativas existentes na literatura para a resolução deste problema
- Apresentar o contexto do problema de alocação na UFC e no SMD
- Apresentar o processo de estudo para elaboração do solução tecnológica para este problema
- Discutir as técnicas e ferramentas utilizadas para a modelagem do sistema
- Apresentar os artefatos da solução proposta

Adicionalmente, os artefatos desenvolvidos neste trabalho são disponibilizados na forma de software livre e assim este TCC se caracteriza como Relatório Técnico + Produto Multimídia.

Este documento é dividido da seguinte forma: o Capítulo 2 apresenta todo o arcabouço teórico necessário para o entendimento dos conceitos abordados neste trabalho, o Capítulo 3 aborda os métodos utilizados para a gerenciamento do desenvolvimento do software, o Capítulo 4 é apresentado em detalhes o processo de gerenciamento e de desenvolvimento da aplicação em questão e por fim o Capítulo 5 apresenta as conclusões do trabalho e também são apresentadas

algumas propostas de trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Nesse capítulo serão apresentadas os principais conceitos utilizados neste trabalho, sendo eles o problema de Timetabling e suas subdivisões, Visualização de Dados e Desenvolvimento Web. A partir da compreensão desses conceitos, poderá ser entendido os aspectos mais importantes que envolvem o desenvolvimento da aplicação Oferta Fácil.

2.1 Timetabling

Apesar de não ser o foco deste trabalho específico, é importante apresentar as descrições formais do problema aqui estudado. Como percebido nos capítulos posteriores, a aplicação Oferta Fácil poderá ser utilizada como elemento de visualização de soluções de alocação para muitas instâncias dos problemas aqui descritos.

De acordo com Wren (1996), Timetabling é um caso especial da atividade de agendamento, onde mostra quando eventos específicos devem ocorrer, não necessariamente implicando uma alocação de recursos. Para tal, também é preciso definir agendamento, e o mesmo autor faz essa definição:

> "Vamos considerar que o objetivo do agendamento é: resolver problemas práticos relativos à alocação, sujeita a restrições, de recursos a objetos colocados no espaço-tempo, usando ou desenvolvendo quaisquer ferramentas que possam ser apropriadas. Os problemas muitas vezes se relacionam com a satisfação de certos objetivos."(WREN, 1996).

Para Bardadym (1995) a dificuldade do problema de agendamento inicia quando a quantidade de recursos, objetos e restrições se torna numerosos demais para que os responsáveis em criar o agendamento/Timetabling possam gerenciar gerando uma quantidade excessiva de trabalho rotineiro como checar se todos os requisitos foram cumpridos, buscar erros e criar outras alternativas aceitáveis.

Esta discussão anterior é bastante ampla e os problemas de Timetabling abrangem diversos outros problemas específicos incluindo *Employee Timetabling*, *Rostering Problems*, *Sports Timetabling* e *Educational Timetabling* (KRISTIANSEN; STIDSEN, 2013). Devido a natureza do problema que este trabalho propõem resolver, somente será abordado o *Educational Timetabling*.

Educational Timetabling “consiste em fixar uma sequência de encontros entre professores e alunos em período preestabelecido de tempo, satisfazendo um conjunto de restrições de

vários tipos" (SCHAERF, 1999). Ele é um problema comum encontrado em todas as instituições educacionais e um dos problemas de Timetabling mais estudados (KRISTIANSEN; STIDSEN, 2013). Devido aos diferentes tipos de instituições envolvidas e de restrições, diversas variações de *Educational Timetabling* foram propostas na literatura. De acordo com Kristiansen e Stidsen (2013), ele é divido em quatro principais categorias:

- *University Course Timetabling* - trata do problema onde a tarefa principal é atribuir aulas de cursos a horários, salas e outros recursos, sujeito a restrições aplicadas para um único aluno.
- *High School Timetabling* - trata da tarefa onde se deve alocar aulas para os horários, professores e salas para satisfazer as restrições. Os alunos são agrupados em turmas e geralmente estão juntos para todas as aulas de sua turma.
- *Examination Timetabling* - deve-se agendar um determinado número de exames para um número limitado de horários, evitando conflitos no calendário de exames de cada aluno e garantir que eles tenham tempo suficiente de preparação para cada exame.
- *Student Sectioning* - tarefa onde deve-se atribuir alunos a seções/classes dos cursos, e não horários, respeitando as solicitações de cada aluno.

Logo, será discutido mais profundamente o *University Course Timetabling* na seção a seguir, visto que é o recorte mais adequado ao problema enfrentado dentro do SMD.

2.1.1 University Course Timetabling

Genericamente, "o problema de *University Course Timetabling* está preocupado com grupos ou turmas de alunos que seguem um determinado caminho ou curso definido que possui eventos associados que exigem alocação de tempo e recursos" (MCCOLLUM, 2006). Como pode ser inferido pelo nome, esse problema afeta principalmente as instituições de ensino superior. Segundo Kristiansen e Stidsen (2013), os tamanhos de turmas e complexidade das estruturas curriculares tornam as possíveis soluções bastante complicadas, fazendo com que sejam necessários múltiplos especialistas dos cursos para encontrar uma solução adequada.

Para McCollum (2006), as soluções devem satisfazer uma série de restrições que podem ser divididas em rígidas e flexíveis, da mesma forma os módulos e eventos associados têm de ser programados de forma a oferecer aos alunos e ao corpo docente o máximo de flexibilidade de escolha e garantir que o espaço de ensino seja usado de forma eficaz. Kristiansen e Stidsen (2013) descreve as restrições rígidas mais comuns sendo: sem conflitos diretos tanto em alunos

quanto professores, não mais de um evento por local ao mesmo tempo, e salas devem satisfazer quaisquer condições exigidas pelo evento. Enquanto as restrições flexíveis mais comuns são: capacidade de sala, horários otimizados e aulas em salas fixas para manter estabilidade da oferta.

Portanto, para McCollum (2006), uma solução de qualquer problema de Timetabling é aquela que satisfaça todas as restrições. Qualquer software que possa chegar nesse resultado é frequentemente vista como uma boa solução, uma solução otimizada não é geralmente o objetivo principal dos administradores universitários. Para universidades com dificuldade pelo grande número de alunos, salas, professores e horários, é comum optar cada vez mais pela automação dessa tarefa para produzir soluções eficientes que satisfaçam essas restrições. Mesmo assim, também é normal nesses casos haver intervenção manual de especialistas.

A seguir, algumas categorias das principais metodologias utilizadas e, trabalhos científicos sobre *University Course Timetabling* (KRISTIANSEN; STIDSEN, 2013):

- *Swarm Intelligence Algorithms*
- *Evolutionary Algorithms*
- *Local Search Algorithms*
- *Graph Coloring Algorithms*
- *Exact Methods*
- *Software systems*

Como o foco do trabalho aqui presente é os métodos manuais, não será feita a discussão acerca dessas metodologias.

Para a solução de qualquer problema de *Educational Timetabling* é necessário um corpo especializado da instituição de ensino para que todas as restrições sejam atendidas de forma suficiente. Esse corpo possui informações mais precisas e completas que dificilmente poderiam ser usadas algoritmos ou devido a sua especificação, ou por serem restrições temporários com uma curta duração. As soluções manuais utilizam o conhecimento mais aprofundando desse corpo para chegar em um resultado mais satisfatório. No curso de Sistemas e Mídias Digitais é usado a seguinte heurística:

1. As disciplinas são inicialmente separadas baseado em suas categorias, unidades curriculares, trilhas e semestres;
2. Cada professor é provisoriamente atribuído de acordo com as necessidades de cada disciplina;
    a) A atribuição deve manter a quantidade homogênea de disciplinas de cada professor,

fazendo exceções em casos específicos;

3. Os horários são provisoriamente atribuídos aos professores e disciplinas;
    a) Os horário são discutidos com os professores para confirmar a disponibilidade;
    b) Caso o horário não seja possível deve-se refazer o passo 3 ou 2 até que todos os horários estejam definidos;
4. salas são atribuídas as disciplinas e seus horários;

Entretanto, de acordo com Schaerf (1999), esta tarefa geralmente requer vários dias de trabalho e, em adição, a solução obtida pode ser não satisfatória em alguns casos. Por exemplo, dependendo do agendamento específico de um par de turmas, um estudante pode não conseguir matrícula neste par de componentes por eles estarem agendados para o mesmo horário. Assim, o conhecimento do especialista pode ajudar a capturar esses elementos mais específicos de diferentes grupos de alunos. Adicionalmente, esse processo pode ser auxiliado por *sistemas de software* o foco particular do presente trabalho, que removem boa parte da carga de trabalho em etapas repetitivas, como a checagem de conflitos de horários ou salas e permitem que o especialista foque exclusivamente em uma melhor visualização dos requisitos específicos que ele leva em consideração para a oferta de turmas.

2.2 Dados tabulares

Para que um corpo administrativo de uma instituição de ensino possa chegar na solução do problema de *Timetabling*, seja por soluções manuais ou não, é necessário que eles possam visualizar todos os recursos e objetos necessários e suas restrições sem que o ato de visualizar se torne mais trabalhoso que o ato de solucionar o problema. Com isso em mente, é importante apresentar as descrições formais de visualização de dados e o método de dados tabulares.

Com base em Khan e Khan (2011), visualização é uma representação gráfica que possa transmitir qualquer ideia de forma compreensível e de fácil entendimento. Seu objetivo principal é analisar grandes quantidades de informações e poder transmiti-las de forma compreensível provendo múltiplos pontos de vistas e níveis de detalhes. Dentro as diferentes técnicas de visualização de informações existentes a Visualização de Dados é uma delas. Também de acordo com Khan e Khan (2011), a Visualização de Dados é o estudo da representação de dados de forma organizada e visualmente agradável para eles poderem ser entendido de forma eficaz.

Das diversas técnicas de representação de dados, para Khan e Khan (2011), a tabela

é uma das mais simples e fácil entendimento, por possuir uma organização centrado em linhas e colunas e como a duas se relacionam. Devido a sua simplicidade e flexibilidade a tabela tem um papel importante nas áreas de pesquisa e análise de dados.

Por exemplo, temos na Figura 1 um modelo simples de tabela. A primeira linha é composta pelos nomes dos elementos que cada coluna representa, por exemplo, a coluna nomeada *Paragraphs* lista todos os parágrafos sendo observados, já a primeira coluna é composta pelos nomes dos elementos que cada linha representa, por exemplo, a linha nomeada *Paragraphs 1* lista todos os valores pertencentes ao elemento *Paragraphs 1*. No entanto, os campos isolados não dão nenhuma informação nova, mas sim a combinação de sua coluna e linha, por exemplo, o campo com o valor 77 localizado na segunda coluna e linha informa que o tamanho do *Paragraphs 1* usando a unidade de medida *C/L Without Spaces* é 77. Outras características importantes são o uso de cores usualmente para organizar elementos em uma mesma categoria como as últimas quatro linhas, além disso, os campos da tabela podem conter mais de um valor, por exemplo, os campos da terceira coluna possuem dois valores o primeiro representando a medida e o segundo uma variação.

Figura 1 – Tabela exemplo

Paragraphs	C/L Without Spaces	C/L With Spaces	C/L in Inches
Paragraphs 1	77	93(±5)	5.25
Paragraphs 2	111	133(±5)	9.13
Paragraphs 3	116	141(±5)	9.5
Paragraphs 4	102	122(±5)	10.74
Paragraph 5	74	87(±5)	4.87
Paragraph 6	96	115(±5)	8.13
Paragraph 7	114	134(±5)	9.4
Paragraph 8	96	111(±5)	10

Fonte: (KHAN; KHAN, 2011)

2.3 Desenvolvimento web

Com a profusão do uso de computadores e dispositivos móveis com as mais diversas características, é importante construir soluções de software que possam ser facilmente utilizadas nessas diferentes plataformas. O desenvolvimento web é um meio de se obter soluções com essa característica.

Podemos caracterizar o desenvolvimento web como a área responsável pelo de-

senvolvimento de sites e aplicações web, podendo ser dividido em duas frontes principais: o Front-end responsável pela interface gráfica do site e o Back-end responsável pela lógica de negócios e interagir com o banco de dados no servidor. Aplicações web são geralmente bastante versáteis tendo um baixo custo, baixo tempo de desenvolvimento e serem bem acessíveis já que não requerem instalação e em sua grande maioria requerem pouco poder de processamento. Além dessas características as aplicações web são bem versáteis em suas arquiteturas, podendo aplicações semelhantes terem padrões distintos.

2.3.1 Arquitetura de software

Arquitetura de software, de acordo com Valente (2020), possui duas definições comuns. A primeira sendo mais prática focando nos elementos de "maior tamanho"e maior relevância do sistema como pacotes, componentes, módulos ou serviços. Já a segunda é mais conceitual focando nas decisões que tomadas antes de iniciar o projeto já decisões como qual linguagem de programação usar ou qual banco de dados será usado impactará quais modulas essências para o sistema serão usados. Esta primeira trata do que chamamos comumente de arquitetura de software.

Valente (2020) apresenta diversos estudos e formas de se construir o projeto de arquitetura de um software, a partir das características que se deseja. Como exemplos importantes para o desenvolvimento web, o autor cita Arquitetura em Camadas e Arquitetura MVC como sendo uns dos mais comuns padrões.

A Arquitetura em Camadas é dividida em camadas de forma hierárquica, cada camada só pode usar serviços com camadas imediatamente abaixo delas na hierarquia. Uma variação muito comum é a Arquitetura em Três Camadas onde a arquitetura normalmente é distribuída, ou seja, cada camada é executada em locais diferentes, a camada de apresentação é responsável por toda parte da aplicação que o cliente pode interagir diretamente e geralmente fica na máquina dos clientes, camada de aplicação é responsável pelas regras de negócios do sistema e geralmente é executada em um servidor e por fim o banco de dados responsável por armazenar os dados do sistema.

A Arquitetura MVC ou Model-View-Controller define que as classes do sistema devem ser organizadas em três grupos : visão, controladores e modelo. A visão é responsável pela interface gráfica do sistema incluindo qualquer elemento visual, os controladores são responsáveis por tratar e interpretar eventos gerados diretamente ou indiretamente pelos usuários

como apertar um botão e por ultimo o modelo é responsável por armazenar os dados da aplicação que podem ser manipulados ou interagidos tanto pelos controladores como pela visão. Muitos dos frameworks atuais utilizam esse método como uma base ou possuem similaridades com ele como os JavaScript Framework.

2.3.2 JavaScript Framework

Desenvolvimento web front-end utiliza primariamente JavaScript geralmente junto com HTML e CSS3, porém usar essas linguagens de forma pura para a construção de sistemas web pode ser um processo lento e complexo. Assim tem tido uma emergência de JavaScript Frameworks devido a necessidade de facilitar, gerenciar melhor e reduzir a complexidade do desenvolvimento de aplicações web.

Segundo Mariano (2017) JavaScript Framework é uma determinada estrutura de como o código deve ser escrito. É um conjunto de funções e ferramentas que facilitam muito o desenvolvimento de código JavaScript compatível com vários navegadores reduzindo muito o custo e o tempo de desenvolvimento.

Atualmente existe diversos frameworks em funcionamento e sendo atualizados, dentro eles estão Angular, React, Vue.js, jQuery e entre outros. Cada um possui seus pontos negativos, positivos e quais necessidades eles melhores se encaixam. A Figura 2 apresenta uma descrição resumida dos frameworks e algumas de suas qualidades e problemas.

Figura 2 – Tabela comparativa de JavaScript Frameworks

Framework	Descrição	Comparação
jQuery Jquery	Uma biblioteca JavaScript open source, foi desenvolvida para facilitar manipular o HTML e os elementos DOM. Funciona simplificando o Javascript tornando algumas funções mais fáceis e rápidas de serem utilizadas.	Permite uma boa compatibilidade entre os navegadores. Por ser uma biblioteca já antiga possui diversos plug-ins e extensões. No entanto necessita de outras linguagens para ter acesso aos dados do servidor. Além de sofrer de problemas de performace em aplicações mais complexas.
 React	Uma biblioteca JavaScript open source apoiado pelo Facebook utilizando o JSX que combina JavaScript com HTML. Muito utilizada para criar interfaces de usuários.	O JSX permite escrever toda a página em um único arquivo JavaScript. É fácil de utilizar e possui uma comunidade muito ativa. No entanto não possui uma boa documentação e por ter um padrão flexível isso pode deixar o projeto pouco escalável.
 Vue.js	Um framework de open source criado por Evan You. Foi desenvolvido para ser progressivo permitindo ser adicionado em qualquer ponto a uma aplicação.	Ele é leve e rápido além de ter uma curva de aprendizado baixo. Além de poder ser migrado para outros frameworks com facilidade. Porém sua flexibilidade pode causar problemas de padronização de código.
 Angular	Um framework de open source criado pelo Google baseado em TypeScript. Foi desenvolvido como uma reescritura do antigo AngularJs.	Arquitetura baseada em components o que facilita a reutilização de código e o teste do mesmo. TypeScript fornece mais vantagens em questão de escabilidade e organização além de maior legibilidade do código. No entanto possui uma complexidade considerável o que não é ajudado pelas várias versões disponível com sintaxes e estruturas diferentes.

Fonte: elaborado pelo autor(2022)

3 METODOLOGIA

O processo de sair de um problema definido por um cliente para chegar em uma solução específica de software é bastante longo e deve ser percorrido de forma cuidadosa para que todos os envolvidos possam estar satisfeitos ao final do processo, bem como todas as restrições de tempo e financiamento também possam ser satisfeitas. Assim, para guiar o processo de definição do problema enfrentado e de desenvolvimento do protótipo proposto pelo Oferta Fácil, foram adotados alguns dos conceitos do *Design Thinking* e do SCRUM. A seguir, discutiremos muitos dos conceitos existentes nesses dois métodos. Apesar do conteúdo deste capitulo ser uma revisão bibliográfica similar ao capítulo anterior, os conceitos apresentados foram usadas diretamente no processo de desenvolvimento do protótipo Oferta Fácil.

3.1 Design Thinking

As definições de *Design*, *Design Thinking* e suas etapas usadas a seguir são específicas de Ambrose e Harris (2016). O principal motivo para tal escolha é a forma o autor separou as etapas do *Design Thinking* em suas formas mais básicas permitindo um entendimento melhor do processo, mas também uma flexibilização maior do mesmo.

O *Design* é um processo interativo que tem como objetivo transformar requisitos e ideias em um produto finalizado ou solução de projeto e *Design Thinking* está presente em cada etapa desse processo (AMBROSE; HARRIS, 2016). Esse objetivo é alcançado buscando diversos ângulos e perspectivas para solução de problemas, priorizando o trabalho colaborativo em equipes multidisciplinares em busca de soluções inovadoras que possam entender e atender o cliente em potencial.

Embora o processo de Design Thinking seja muitas vezes linear ele frequentemente envolve revisitar etapas anteriores para serem retrabalhadas à medida que o projeto evolui e novas informações são adquiridas. Segundo Ambrose e Harris (2016), o Design Thinking é composto por sete etapas:

1. **Definição** - O problema de design e o público-alvo são definidos e é determinado o que é necessário para o sucesso do projeto;
2. **Pesquisa** - São realizadas pesquisas quantitativas e ou qualitativas sobre o projeto para adquirir informações sobre o público-alvo e possíveis obstáculos;
3. **Ideação** - Também é conhecida como a fase do brainstorming. As informações adquiridas

pelas pesquisas são analisadas e usadas para desenvolver novas ideias e soluções para o problema;

4. **Prototipação** - Das ideias e soluções geradas são selecionadas as mais promissoras para receberem protótipos e serem revisadas por grupos de usuários;
5. **Seleção** - É selecionado uma das ideias ou soluções que foram revisadas que mais se aproximam de resolver o problema do projeto;
6. **Implementação** - Represento o desenvolvimento da solução do projeto e sua entrega final ao cliente e ao público-alvo;
7. **Aprendizagem** - O designer recebe feedback do cliente e do público-alvo de como o produto foi recebido pelo público-alvo, quais suas qualidade e problemas. Ajudando os designers a melhorar seu desempenho;

Cada umas destas etapas recebe como insumos o resultado da etapa anterior e repassa seus resultados para a próxima etapa. Esta característica linear de tarefas pode ser visualizada na Figura 3.

Figura 3 – Sete etapas do design thinking

Fonte: (AMBROSE; HARRIS, 2016)

Caso se opte por uma solução de software, deve-se seguir um processo de engenharia de software na etapa de implementação.

3.2 Engenharia de Software

Valente (2020) define Engenharia de Software como a aplicação de princípios de engenharia de forma orientada e sistemática na construção e desenvolvimento de software. Um desses princípios é a Engenharia de Requisitos responsável por identificar e estudar os requisitos de um sistema.

3.2.1 Requisitos

Requisitos são, de acordo com Valente (2020), definições do que um sistema deve fazer e suas restrições. Eles podem ser separados em dois grupos baseado em suas definições, os **Requisitos Funcionais** são requisitos que abrangem as funcionalidades do sistema e suas características, já os **Requisitos Não-Funcionais** abrangem as restrições do sistema. Por exemplo, em um sistema de agendamento um requisito funcional seria agendar um horário, enquanto um requisito não-funcional seria o tempo para processar cada alteração no agendamento.

No entanto os requisitos sozinhos são somente as instruções de como criar um sistema, ainda é necessário um método de como seguir tais instruções. Para tal a Engenharia de Software prover metodologias que levam a produção de tais softwares, cada uma contendo suas desvantagens e vantagens. Como exemplo temos os Métodos Clássicos e os Métodos Ágeis.

3.2.2 Métodos Ágeis

Os Métodos Ágeis são diversas metodologias baseados em desenvolvimento iterativo, no qual requisitos e soluções evoluem pela colaboração entre equipes auto-organizadas e cross-funcional (PRIKLADNICKI *et al.*, 2014). Porém, elas são vão receber esse nome em 2001, quando o Manifesto Ágil é criado por especialistas da área, muitos responsáveis pelas próprias metodologias. Algumas metodologias e framework que se encaixam nos métodos ágeis e suas diferenças são:

- Programação Extrema (XP) - tem o foco nos aspectos técnicos do desenvolvimento e levando ao extremo boas práticas como: programação em par, testes constantes e refatorações de código.
- Kanban - utiliza pequenos cartões coloridos, ou post-it, para representar as tarefas em um quadro dividido em colunas que representam os estados das tarefas. A equipe responsável é por mover os post-it de acordo com os progressos das tarefas.
- Scrum - possui como principal característica seu ciclo de eventos de duração fixas chamadas de sprint, com duração dependendo do projeto.

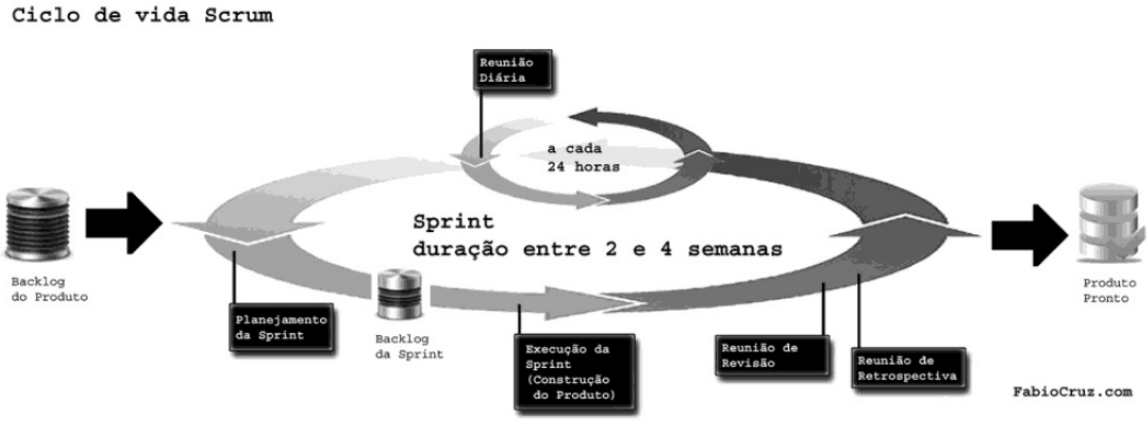
3.2.3 Scrum

O Scrum é um framework para projetos ágeis utilizado para o gerenciamento e desenvolvimento de produtos, com a característica de ser iterativo e incremental. Ele se baseia

em ciclos chamados de Sprint que podem possuir durações de duas a quatro semanas dependendo do projeto. Cada Sprint contém as cerimônias do Scrum que são (CRUZ, 2013):

- As **reuniões de planejamento da Sprint** é nela que toda a Sprint é planejada e o que e como será feito.
- As **reuniões diárias** onde o Time se encontra diariamente para uma reunião curta para fazer um alinhamento do progresso dos membros e dificuldades que eles estejam enfrentando
- A **revisão da Sprint** acontece no final da Sprint onde é realizado a revisão pelo *Product Owner* ou cliente de todos os itens que foram concluídos pelo Time.
- A **retrospectiva da Sprint** ocorre depois da revisão, o Time faz uma avaliação do próprio processo de desenvolvimento e como fazer melhorias e reduzir problemas para a próxima Sprint.

Figura 4 – Ciclo de vida Scrum



Fonte: (CRUZ, 2013)

Esse framework consiste em um time divido em papéis e responsabilidades sendo eles o *Scrummaster* responsável pelo time seguir as práticas e regras do Scrum enquanto remove ou suaviza impedimentos que possam interferir com o time, o *Product Owner* responsável por manter o Backlog do produto e sua integridade além de garantir que o time compreenda o produto e o mantenha de acordo com desejos do cliente e por fim o Time é responsável por transformar o Backlog do produto e seus requisitos no produto final. Os membros do Time não possuem títulos ou cargos e possuem o conhecimento e a habilidade de diversas disciplinas.

O Backlog do produto é o principal artefato do Scrum ele agrega todas as características, funções, requisitos, melhorias e correções a serem feitas ou implementadas ao produto. Outros artefatos são o Backlog da Sprint que consistes dos itens selecionados do backlog do produto para a sprint atual e o Incremento do produto que segundo Prikladnicki *et al.* (2014),

consiste no resultado de tudo produzido durante a sprint. Estes artefatos podem ser visualizada na Figura 4. Oficialmente esses são os únicos artefato do Scrum, porém com o passar do tempo outros artefatos foram adicionados e considerados partes do Scrum sendo eles as histórias e o quadro de tarefas (CRUZ, 2013).

4 DESENVOLVIMENTO

Neste capítulo será mostrado o processo de desenvolvimento da aplicação, passando pelo estágio de pesquisa, a metodologia de gerenciamento aplicada, o processo de desenvolvimento e o framework utilizado. A ideia é apresentar de forma detalhada cada etapa, expondo as vantagens e limitações trazidas pelas ferramentas escolhidas.

4.1 Problema

O Sistemas e Mídias Digitais é um dos vários cursos da Universidade Federal do Ceará (UFC) e como todos os cursos da universidade é necessário que seja criado uma oferta de turmas antes de cada semestre iniciar. No entanto devido ao fato que o sistema de gerenciamento acadêmico da universidade, o Sistema Integrado de Gestão de Atividades Acadêmicas (SIGAA), não possuir um módulo que execute tal função de forma automatizada, a oferta de turmas é efetuada de forma manual pelo conjunto de coordenadores do SMD Diurno e SMD Noturno com auxílio dos colegiados dos cursos.

O curso de Sistemas e Mídias Digitais tem em sua matriz curricular oito semestres para o curso Diurno e nove semestres para o Noturno, sendo os três primeiros semestres com disciplinas obrigatórias fixas e os seguintes possuindo uma mistura de disciplinas obrigatórias, optativas e eletivas, em um sistema de trilhas. Disciplinas obrigatórias e optativas são comuns a todo e qualquer curso de graduação, já as disciplinas eletivas são grupos de disciplinas para os quais o aluno deve cursar um subconjunto delas. Por exemplo, no quarto semestre do SMD são ofertadas turmas de 6 disciplinas eletivas das quais os alunos devem obrigatoriamente cursar 4 delas. O SIGAA reconhece disciplinas obrigatórias e optativas, as eletivas devem ser computadas e visualizadas manualmente por qualquer interessado.

As trilhas são conjuntos de disciplinas optativas que englobam conteúdos de áreas específica, funcionando como um guia para auxiliar os alunos a decidirem quais disciplinas devem escolher para poder se especializar em uma área específica. Por exemplo, a trilha de Sistemas engloba muitas disciplinas relacionadas à construção de sistemas multimídias que podem ser cursadas em diferentes semestres do curso. Esse sistema não é restrito, podendo um aluno escolher disciplinas de diferentes trilhas. O SIGAA também não reconhece esse sistema de trilhas, a comunicação e visualização das mesmas deve ser feita de forma manual por qualquer interessado.

Além disso, o bloco onde o curso é localizado possui 5 salas, 1 ateliê e 6 laboratórios nos quais devem atender as necessidades de cada turma, onde aproximadamente 40 professores dão aulas para pouco menos 90 turmas, divididas nos turnos matutino, vespertino e noturno. A oferta de uma determinada turma é constituída por uma disciplina e um ou mais professores associados.

A partir de primeiras reuniões de *Briefing*, verificou-se que muito das informações listadas no problema acima descrito são resumidas e resolvidas através de um conjunto de tabelas construídas na ferramenta de planilhas *Google Sheets*, conforme apresentado na Figura 5. A planilha mostrada na figura é responsável pela criação da oferta de turmas, ela é divida em tabelas menores que representam uma turma, no caso mostrado pela figura 5 a Turma A e a Turma B do 1º semestre, cada tabela menor possui colunas que representam os dias da semana e linhas que representam períodos de tempo sendo AB e CD, respectivamente, 14h às 16h e 16h às 18h. Cada campo de interseção entre uma linha e coluna representa uma aula e nele está contido as informações dessa aula como: nome da disciplina, nome do(a) professor(a), capacidade máxima de alunos e nome da sala. Além disso, cada campo possui cores diferentes de seus vizinhos, essa colorização não somente funciona como uma formar de distinguir melhor individualmente cada aulas, mas também para distinguir qual categoria as disciplinas dessas aulas pertencem.

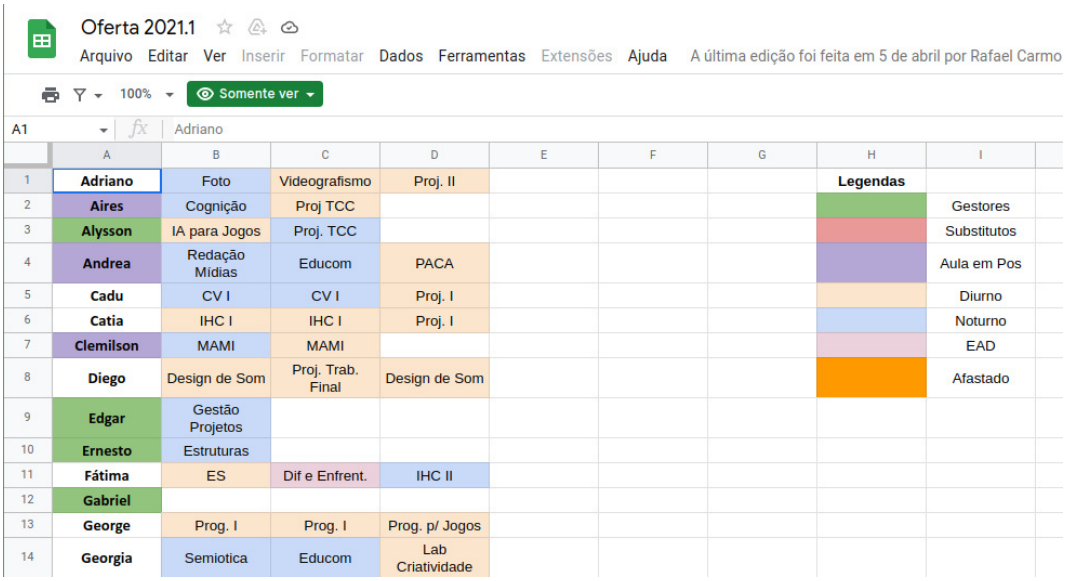
Figura 5 – Criação da oferta de turmas manualmente

Fonte: elaborado pelo autor(2022)

Além da planilha responsável pela oferta são construídas mais duas planilha responsáveis por realizar a análise de dados. A primeira pode ser observada na Figura 6, possuindo somente uma tabela ela é responsável por visualizar de forma mais concisa todas as disciplinas que foram adicionadas na oferta de turmas e quais professores elas foram designadas. Diferentes do padrão ela não possui colunas nomeadas, porém seus nomes podem ser inferidos sendo

a primeira coluna os professores e as demais as disciplinas associadas a tal professor. Além disso, ela possui legendas responsáveis por sinalizar qual categoria ou estado os professores e as disciplinas pertencem.

Figura 6 – Planilha de todas as disciplinas designadas a todos os professores



Oferta 2021.1

Arquivo Editar Ver Inserir Formatar Dados Ferramentas Extensões Ajuda A última edição foi feita em 5 de abril por Rafael Carmo

100% Somente ver

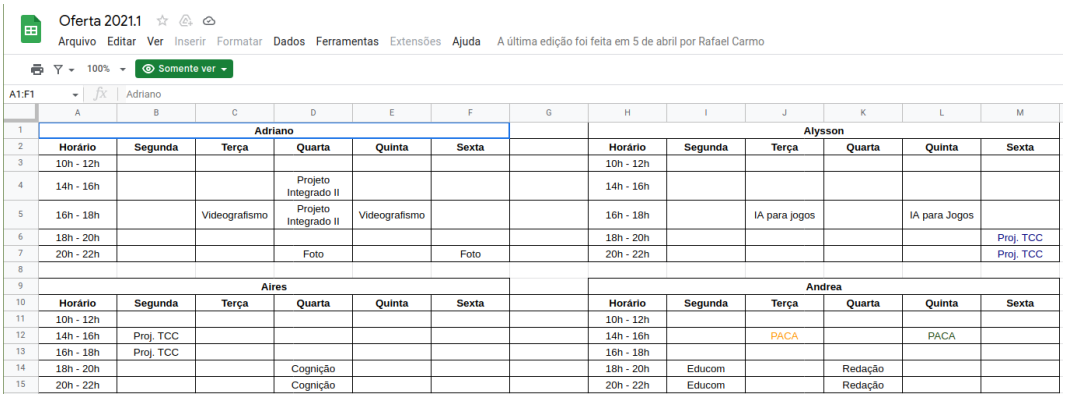
A1 Adriano

	A	B	C	D	H	I
1	Adriano	Foto	Videografismo	Proj. II	Legendas	
2	Aires	Cognição	Proj TCC			Gestores
3	Alysson	IA para Jogos	Proj. TCC			Substitutos
4	Andrea	Redação Mídias	Educom	PACA		Aula em Pos
5	Cadu	CV I	CV I	Proj. I		Diurno
6	Catia	IHC I	IHC I	Proj. I		Noturno
7	Clemilson	MAMI	MAMI			EAD
8	Diego	Design de Som	Proj. Trab. Final	Design de Som		Afastado
9	Edgar	Gestão Projetos				
10	Ernesto	Estruturas				
11	Fátima	ES	Dif e Enfrent.	IHC II		
12	Gabriel					
13	George	Prog. I	Prog. I	Prog. p/ Jogos		
14	Georgia	Semiotica	Educom	Lab Criatividade		

Fonte: elaborado pelo autor(2022)

A segunda planilha é um conjunto de tabelas como pode ser observada na Figura 7, ela é responsável por visualizar de forma mais concisa todos os professores que foram adicionadas na oferta de turmas e quais são os horários das disciplinas que eles foram designados. Cada tabela individual representa o cronograma de uma professor e de forma similar a oferta de turmas elas são dividas em colunas que representam os dias da semana e linhas que representam períodos de tempo e cada campo de interseção entre uma linha e coluna representa o período na qual a aula daquela disciplina será ministrada.

Figura 7 – Planilha de todos os horários de todos os professores



Oferta 2021.1

Arquivo Editar Ver Inserir Formatar Dados Ferramentas Extensões Ajuda A última edição foi feita em 5 de abril por Rafael Carmo

100% Somente ver

A1:F1 Adriano

Adriano							Alysson					
Horário	Segunda	Terça	Quarta	Quinta	Sexta		Horário	Segunda	Terça	Quarta	Quinta	Sexta
10h - 12h							10h - 12h					
14h - 16h			Projeto Integrado II				14h - 16h					
16h - 18h		Videografismo	Projeto Integrado II	Videografismo			16h - 18h		IA para jogos		IA para Jogos	
18h - 20h							18h - 20h					Proj. TCC
20h - 22h			Foto		Foto		20h - 22h					Proj. TCC

Aires							Andrea					
Horário	Segunda	Terça	Quarta	Quinta	Sexta		Horário	Segunda	Terça	Quarta	Quinta	Sexta
10h - 12h							10h - 12h					
14h - 16h	Proj. TCC						14h - 16h		PACA		PACA	
16h - 18h	Proj. TCC						16h - 18h					
18h - 20h			Cognição				18h - 20h	Educom		Redação		
20h - 22h			Cognição				20h - 22h	Educom		Redação		

Fonte: elaborado pelo autor(2022)

O problema surge da combinação de que a oferta tem que ser feita de forma manual e da grande quantidade de recursos, objetos e restrições que devem ser atendidas ao mesmo tempo que reduza a insatisfação de ambos corpos docentes e discentes. O que resulta com que os responsáveis pela oferta utilizem ferramentas e técnicas próprias, muitas vezes inadequadas, para executar esta tarefa a cada semestre letivo. Atualmente ela é feita usando planilhas online, ou seja, disciplinas, professores e salas devem ser adicionadas ou removidas manualmente da planilha caso seja necessária alguma alteração. Além disso, as análises de dados são realizadas também de forma manual em diferentes planilhas como observado nas Figuras 6 e 7, por consequências qualquer alteração feita na planilha de ofertas não será automaticamente transferida às planinhas de análise de dados e vice-versa, assim resultando em erros e inconsistências entre as planilhas.

4.2 Produtos concorrentes

Com o principal problema e suas peculiaridades propriamente definidos foi realizado uma pesquisa por produtos e aplicações que pudessem se adequar como uma solução para o problema, com o objetivo de observar seus principais pontos negativos, positivos e como eles lidam com certas dificuldades e processos. Foram selecionados quatro aplicações para serem observadas sendo duas sendo software proprietário e duas sendo software livre.

4.2.1 Unitime

Unitime é um sistema colaborativo web open source desenvolvido por membros de universidades norte-americanas e europeias com o objetivo de auxiliar no gerenciamento de horários das universidades ou em pesquisas realizadas na área. O sistema funciona através de uma combinação de ferramentas de automação e manuais, que não somente para gerenciamento de horários de aulas, mas também de exames e eventos. Para realizar o processo de automação o usuário deve criar múltiplas restrições e condições para todos os professores, disciplinas, salas e horários, assim o sistema pode analisá-las e desenvolver uma solução que as atenda. As ferramentas manuais permitem que o usuário criei sua própria solução ou que altere uma criada pelo sistema através de uma interface interativa.

Sua principal vantagem é não somente sua flexibilidade e complexidade de ferramentas e funções, mas também sua longevidade sendo um projeto que existe a mais de uma década sofrendo constante atualizações e melhorias. No entanto sua variedade acaba por se tornar uma

desvantagem, pois suas muitas funcionalidades estão escondidas em interfaces arcaicas e pouco amigáveis ao usuário. Por fim, apesar de ser uma aplicação web não possui um servidor dedicado tendo assim a necessidade do usuário ter um servidor próprio ou ser hospedar localmente o sistema.

4.2.2 aSc Horários

ascHorarios é um sistema desktop de software proprietário com a função de gerenciar grades de horários escolares de ensino básico, desenvolvido e distribuído pela aSc Applied Software Consultants. O sistema funciona através de uma combinação de ferramentas de automação e manuais, que fazem gerenciamento de horários de aulas. Como descrito no sistema anterior para realizar o processo de automação o usuário deve criar múltiplas restrições e condições, assim o sistema pode analisá-las e desenvolver uma solução que as atenda. As ferramentas manuais permitem que o usuário criei sua própria solução ou que altere uma criada pelo sistema através de uma interface interativa.

Sua principal vantagem é sua interface, sendo simples e amigável permitindo que usuário possam utilizar a aplicação em questão de minutos, além de ser bastante acessível já que ela possui extensões que permitem o acesso via web e mobile. No entanto ela não é tão robusta quanto a Unitime na quantidade de funções ou complexidade e por ser uma software proprietário é necessário pagar a licença para não somente ter acesso completo a ela, mas acesso as extensões. Os valores variam entre 120 a 1995 dólares, porém a versão gratuita de teste possui a função desbloqueadas com exceção da exportação da solução e relatórios são emitidos com marca d'água.

4.2.3 FET

FET é um sistema desktop multiplataforma de software livre com o objetivo de auxiliar no gerenciamento de horários de instituições de ensino sendo elas de ensino superior ou de ensino básico. O sistema funciona através primariamente do uso de ferramentas de automação, que não somente para gerenciamento de horários de aulas, mas também de exames e eventos. Como nos outros sistemas citados ele utiliza uma série de restrições e condições atribuídas pelo usuário para que o processo de automação possa chegar em uma solução. Apesar de no ser o foco do sistema ele possui algumas ferramentas manuais, mas são limitas em sua utilização.

Sua principal vantagem é sua quantidade, flexibilidade e complexidade de ferramen-

tas e funções para criar as restrições e condições necessárias para o algoritmo de automação. No entanto como a aplicação Unitime sua vantagem é também sua principal desvantagem, a quantidade de elementos somadas com uma interface gráfico simplista torna a aplicação pouco amigável para usuários iniciantes ou mais experientes.

4.2.4 Centralized academic scheduling software

Centralized academic scheduling software é um sistema desktop de software proprietário com a função de gerenciar grades de horários de instituições de ensino superior, desenvolvido e distribuído pela Lantiv. O sistema funciona através primariamente do uso de ferramentas manuais, que não somente fazem o gerenciamento de horários de aulas, mas também de exames e eventos. Diferentes dos outros sistemas citados ele possui poucos elementos de automação sendo a maioria deles somente para preenchimento rápido de informações. Seu principal foco é o preenchimento manual de um cronograma em uma interface interativa, possuindo diversas formas de visualização e uma flexibilidade maior de horários.

Sua principal vantagem é sua interface, sendo complexa, mas amigável permitindo que usuário possam utilizar a aplicação sem muitas dificuldades. Por ser uma software proprietário é necessário pagar um sistema de inscrição no valor de 49 dólares para ter acesso completo a ela.No entanto a versão de teste pode ser usada, mas ela não aceita importação de novos dados ou salvar o arquivo feito.

4.3 Requisitos do sistema

Com as informações coletadas de como outras aplicações resolveram o problema principal foi iniciado a elicitação de requisitos. Em reuniões realizadas com o cliente, Professor Dr. Rafael Augusto Ferreira do Carmo, foi determinado os elementos mais importantes para a solução do problema, como o problema era solucionado no curso de sistemas e mídias digitais e o conceito base do protótipo. Desse modo, os requisitos funcionais foram sendo consolidados ao longo das reuniões.

4.3.1 Requisitos Funcionais

No final da consolidação dos requisitos funcionais os seguintes foram definidos e podem ser encontrados no Apêndice A.

4.3.2 Especificação de requisitos do sistema

As definições dos requisitos funcionais apesar dar uma explicação do que é o requisito não aborda por completo como esse requisito influencia o sistema e os usuários. Desse modo, será listado as especificações de cada requisito onde entrarão em mais detalhes sobre seu funcionamento. As especificações podem ser encontradas no Apêndice B.

Com os requisitos criados e especificados é ainda necessário organizá-los ao nível de complexidade e importância além dividi-los em tarefas menores de forma que eles possam permitir uma progressão mais linear do desenvolvimento. Para tal é necessário um método ou modelo de gerenciamento que se adéque aos requisitos e suas especificações.

4.3.3 Regras de negócio

O termo regras de negócio refere-se às diretrizes que definem ou restringem ações, mostrando como as operações devem ser conduzidas e se há algum limite nessa aplicação.

As regras de negócio são diretrizes que tem como função restringir e especificar como certas funções ou aspectos de um sistema devem funcionar. Assim, foram definidas as seguintes regras de negócio encontrada no Apêndice C.

4.4 Gerenciamento de projeto

Para que qualquer projeto prossiga de forma eficiente é essencial ter um sistema de gerenciamento de projeto que se adéque a equipe e ao projeto. Assim foi escolhido o Scrum, segundo Cruz (2013), ele é um framework ágil que propõe um auto-gerenciamento dinâmico, versátil, altamente adaptável e muito eficiente na entrega de produtos. Para o autor seu ponto mais forte é também seu mais fraco, devido ao foco em desenvolvimento de produto outras áreas do gerenciamento de projeto ficam ausentes como: custos, orçamento, tempo e aquisições. Porém, devido a natureza do projeto ser sem fins lucrativos o Scrum se torna ideal para o projeto. Buscou-se uma ferramenta online que satisfizesse as condições do Scrum e pudesse armazenar o projeto, assim foi optado o GitHub para gerenciamento do projeto.

4.4.1 GitHub

O GitHub é uma plataforma de hospedagem de código-fonte e arquivos com controle de versão usando o Git. Nele que esta, contido e salvo o código-fonte da aplicação e o registro das antigas versões já criadas e que deve ficar disponível ao público assim que finalizada. Ele possui outras funcionalidades para auxiliar no desenvolvimento de projetos, entre elas é o projects(classic) uma ferramenta de gerenciamento de projeto com base no sistema Kanban, porém ele possui opções e elementos que permitem modificar a ferramenta para acomodar as especificações do Scrum. As tarefas podem ser criadas através de *issues* como observado na Figura 8 ou diretamente no projects(classic) e são representadas no formato de um cartão. O progresso da tarefa é indicado pelo avanço do cartão entre as listas do quadro, cada lista representando um estágio da tarefa. Os cartões permitem uma maior organização e visibilidades das tarefas garantindo assim que o Scrum Master tenha acesso fácil as informações da sprint.

Figura 8 – Requisitos no GitHub

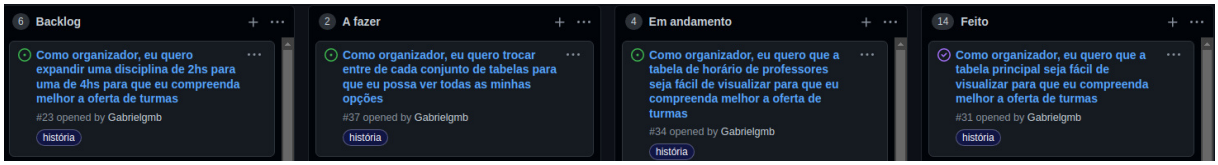
Fonte: elaborado pelo autor(2022)

Assim, para o gerenciamento das tarefas relacionadas ao desenvolvimento do apli-

cativo apresentado neste trabalho, foi utilizada a organização apresentada na figura 1. No caso, o quadro é composto por quatro listas de tarefas, cada lista possuindo um nome indicando o progresso das tarefas:

1. A lista “Backlog” é o Product Backlog do Scrum.
2. A lista “A fazer” possui os cartões que ainda não foram inicializados durante a sprint.
3. A lista “Em andamento” possui os cartões que estão em andamento durante a sprint.
4. A lista “Feito” contém os cartões que foram finalizados.

Figura 9 – Organização do GitHub projects



Fonte: elaborado pelo autor(2022)

O Product Backlog é os requisitos do produto a serem entregue, bem como todo o entendimento necessário para atender aos requisitos, produzir funcionalidades ou produtos e, por fim entregar o produto (CRUZ, 2013). De acordo com o autor, apesar de não oficialmente fazer parte do Scrum as histórias são artefatos comumente utilizados no Scrum, elas consistem em requisitos pequenos ou solicitações escritas da perspectiva de um usuário final. Segue no Apêndice D a lista de todas as histórias criados no Backlog.

4.5 Processo de desenvolvimento

Considerando os requisitos estabelecidos, tarefas propostas e as informações adquiridas nas entrevistas será desenvolvido uma aplicação web composta de um sistema de tabelas interativos, usando de sua praticidade e acessibilidade de não necessitar de instalação e poder ser acessado em qualquer local. Será desenvolvido um protótipo usando a tecnologia Angular 12.2.16 e o angular material para o desenvolvimento, o uso de banco NoSQL para armazenar os dados e a arquitetura de software com base no modelo MVC (Model-View-Controller).

4.5.1 Arquitetura

Na Arquitetura MVC ou Model-View-Controller as classes do sistema são organizadas em três grupos : visão, controladores e modelo. A visão é responsável pela interface gráfica do sistema, os controladores são responsáveis por tratar e interpretar eventos e por ultimo

o modelo é responsável por armazenar os dados da aplicação que podem ser manipulados ou interagidos. Segundo Valente (2020) os aplicativo de página única apresentam uma arquitetura similar com o MVC, são aplicações web que apesar do que o nome sugere não possuem somente uma página e sim somente uma página contém toda a aplicação. O principio é que ao acessar a página todo o código já baixado permitindo a troca de visualização sem delay ou carregamento sendo mais próximo de uma aplicação desktop do que uma web. Frameworks como o Angular utilizam a arquitetura de página única.

4.5.2 Angular

Angular é framework de open source criado pelo Google desenvolvido como uma reescritura do antigo AngularJs. Difere do seu antecessor pelo uso de TypeScript, um superconjunto de JavaScript que adiciona elementos novos a linguagem como tipagem de variáveis facilitando validação de código. Por ser um superconjunto qualquer código em JavaScript pode ser executado em TypeScript sem necessitar de alterações. A segunda mudança é na arquitetura padrão, enquanto AngularJs favorece o padrão model-view-controller (MVC) o Angular se utiliza de componentes. Angular material uma biblioteca de componentes do Angular, responsável por auxiliar e acelerar o desenvolvimento da interface de usuário de aplicações. Uma das vantagens do framework é o uso de TypeScript tanto para criar a interface gráfico quando para executar as regras de negócio e acessar o servidor, por isso é recomendo a utilização de bancos não relacionais ou NoSQL

4.5.3 Banco NoSQL

Banco NoSQL não possuem uma definição precisa, o termo é geralmente usado para bancos que não possuem SQL. "Atuam sem um esquema, permitindo que sejam adicionados, livremente, campos aos registros do banco de dados, sem ter de definir primeiro quaisquer mudanças na estrutura"(SADALAGE; FOWLER, 2019). O sistema NoSQL se encaixa bem com os conceitos do Design Thinking e do SCRUM, onde o projeto está em constante mudança e restruturação. Permitindo que mudanças simples possam ser feitas sem que toda a arquitetura tenha que ser remodelada.

4.5.4 Criando projeto

Com as principais tecnologias já definidas nas subseções anteriores é hora de iniciar o projeto. Tendo Node.js já instalado no sistema operacional é necessário instalar o Angular CLI, para isso é necessário no terminal o seguinte comando no prompt de comando:

```
npm install -g @angular/cli
```

Código-fonte 1 – Instalação do framework Angular

Com o Angular e suas dependências instaladas, já é possível criar um projeto Angular com o comando abaixo, sendo que o comando após o comando "new"é o nome da aplicação:

```
ng new smd-management
```

Código-fonte 2 – Criando um novo projeto em angular

Após a execução do comando acima, é apresentado o seguinte menu com opções de configurações iniciais do projeto:

```
Would you like to add Angular routing? (y/N)
? Which stylesheet format would you like to use? (Use arrow keys)
  CSS
  SCSS  [ https://sass-lang.com/documentation/syntaxscss]
  Sass  [ https://sass-lang.com/documentation/syntaxthe-indented-
   syntax]
  Less  [ http://lesscss.org]
```

Código-fonte 3 – Configurações iniciais de criação de um projeto angular

Essa etapa apesar de ser a configuração do projeto não é permanente podendo ser alterado no decorrer do desenvolvimento. A primeira pergunta é sobre o sistema de routing do Angular, se respondido sim, será criado o arquivo de routing padrão responsável por identificar qual página deve ser carregada e seus nomes, se for respondido não o arquivo não será criado ficando por responsabilidade do desenvolvedor em criar o seu. A segundo pergunta é sobre qual formato de arquivo de estilização deve ser usado, todos tem como base o CSS, mas possuem propriedades extras e pequenas diferenças de formatação.

Foi decidido devido à simplicidade das páginas por adicionar o sistema de routing e foi optado pelo SCSS.

Após iniciar o projeto será feito a última etapa da configuração que é instalar o angular material. Para instalar basta ir para o diretório raiz e executar o comando abaixo:

```
ng add @angular/material
```

Código-fonte 4 – Adicionar a biblioteca de componentes de interface de usuário do Angular

Após confirmar a instalação, será feito algumas perguntas sobre o que deve ser instalado ou não. Com as perguntas respondidas a última etapa é iniciar a aplicação. Para executar a aplicação basta ir para o diretório raiz e executar o comando abaixo:

```
ng serve
```

Código-fonte 5 – Iniciando o servidor local do Angular

Feito isso, o Angular iniciará um servidor de desenvolvimento local para visualização do projeto e um registro de logs no terminal onde poderá ser ver mensagens de aviso e erro. Um dos benefícios do servidor local é a habilidade de poder desenvolver em tempo real já que todas as alterações feitas são automaticamente mostradas no servidor.

4.5.5 Estágios iniciais do projeto

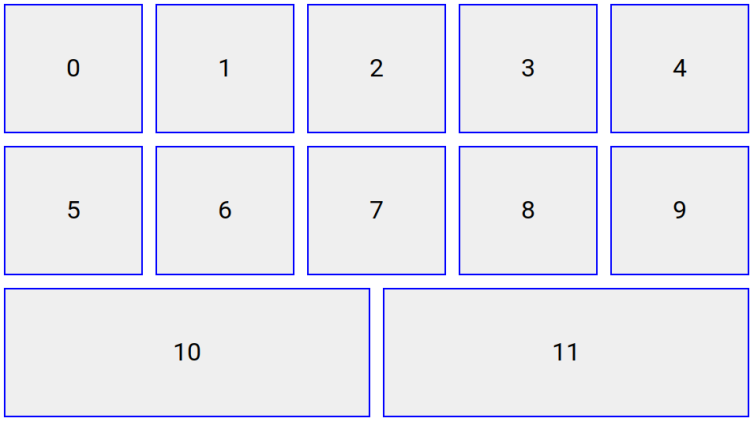
Aqui será relatado o funcionamento e as ideias por trás dos primeiros protótipos e por qual motivo eles foram descartados ou modificados.

4.5.5.1 Tabela interativa

Para criar a tabela interativa será usado um sistema de permutação de blocos, onde você poderia mover blocos em um grid e permutar blocos entre se como observado na Figura 10. Para atingir esse resultado será usado o component *drag-drop* do angular material, esse component permitir interagir com elementos da tela os segurando e arrastando, ou os acomodando em uma lista.

O primeiro protótipo era de uma lista de disciplinas localizadas em uma sidebar que poderiam ser arrastadas para a tabela interativa, na tabela eles poderiam se mover livremente nos

Figura 10 – Print do teste inicial



Fonte: elaborado pelo autor(2022)

blocos da tabela, cada bloco representa um dia da semana e um horário específico. No entanto, esse protótipo foi descartado antes de ser produzido, era impossível manter a consistências da tabela enquanto se movia as disciplinas e era necessário usar a posição das disciplinas nos array como forma de identificação.

O segundo protótipo ainda mantinha uma lista de disciplinas localizadas em uma sidebar, mas ao invés de uma lista sendo usada para criar a tabela, cada bloco da tabela equivale a uma lista. Isso aumenta a estabilidade da tabela e fica mais simples de saber que disciplina esta sendo trocada. No entanto, depois de alguns testes essa versão foi descartada, a sidebar ainda estava causando inconsistência na tabela ao ser adicionado as disciplinas e por ser muito longa, acontecia muita perda de tempo ao procurar a disciplina correta.

O terceiro protótipo manteve cada bloco da tabela equivalente a uma lista. Mas ao invés de ter uma lista única de disciplinas, cada semestre teria uma lista de disciplinas exclusivas daquele semestre. Além disso, a função de arrastar a disciplinas da lista e adicionar a tabela foi alterada para um sistema de clique onde você seleciona a disciplina que você quer adicionar e clica no bloco da tabela. Esse é o sistema base que continuará a ser usado.

4.5.5.2 Serviço e Subject

Com o sistema base da tabela interativa feito, era necessário povoar o sistema com os dados, para isso optei por usar os serviços do angular. Serviço consiste em uma classe que toda a aplicação pode acessar, nele conterá as funções, valores ou característica necessárias para o proposito do serviço. Foi criado dois serviços para o protótipo.

- TableService: responsável por todas as tabelas do protótipo, sua principal função é transformar as informações em tabelas e enviar tais tabelas em formato de objeto para as páginas.

Também é responsável por processar as mudanças feitas pelo usuário na tabela interativa.

- DataService: responsável por adquirir as informações básicas do sistema como: lista de professores, lista de disciplinas, lista de salas e lista de Turmas. E enviar elas para as páginas quando solicitado.

Para as páginas terem uma forma mais consistente e segura de obter o objeto da tabela quando o TableService terminar de processar as mudanças foi utilizado um RxJS Subject para tal. O RxJS Subject é um tipo especial de Observable que permite que valores sejam enviados para múltiplos Observadores. Observable no Angular é um recurso que fornece suporte para a entrega de mensagens entre diferentes partes da aplicação de página única. Para poder usá-lo a pagina deve se inscrever no subject, assim toda vez que for publicado uma nova atualização no subject todas as páginas inscritas receberão a notificação.

O protótipo possui um Subject, o tableSubject, ele é criado dentro do TableService e a página grid se inscreve nele. Quando o protótipo é iniciado ou a tabela sofre uma mudança as funções do serviço são disparadas e criam o objeto contendo as tabelas, ao finalizar o processo a mudança é publica no tableSubject e a página grid recebe a notificação da modificação e puxa o objeto do tableSubject atualizando assim o HTML da página com as novas informações.

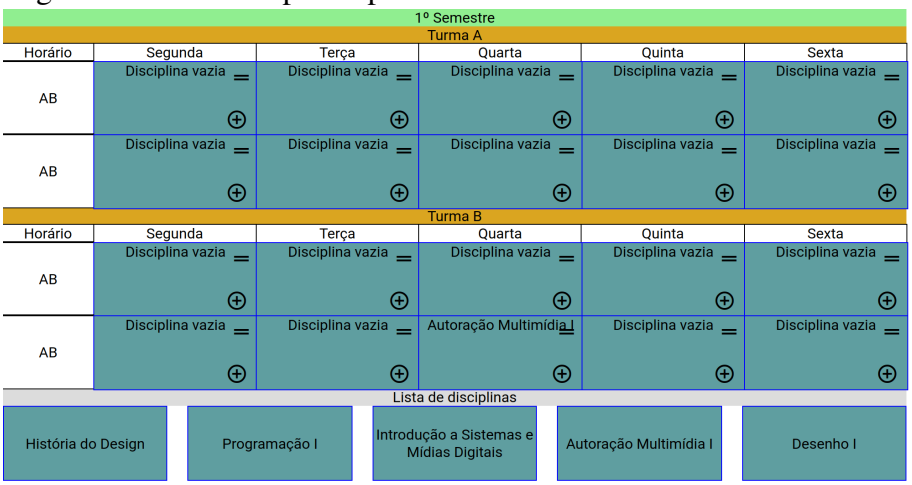
4.5.5.3 Array da tabela

Com o sistema base da tabela e o seu método de comunicação estabelecidos é necessário desenvolver o formato da informação a ser transmitida. Para poder gerar a tabela interativa na página é necessário um objeto contendo todas as informações necessárias: os semestres, as turmas, as disciplinas disponíveis e os horários disponíveis. Porém, como pela forma que a tabela é gerada no html é necessário que esses valores estejam dentro de arrays.

O primeiro protótipo do objeto era bem direto, foi criado uma array tridimensional. A primeira dimensão consistia na lista de semestres, cada semestre continha suas informações essenciais, todas as turmas e disciplinas nele contido. A próxima dimensão, o array de turmas, continha suas informações essenciais e a lista de horários. E por fim a última dimensão, o array de horários, continha suas informações essenciais. Esse método servia tanto para gerar na tabela na página, como pode observado na Figura 11, quanto para guardar as mudanças feitas pelos usuários. Porém, esse método se tornou problemático devido à quantidade de arrays não eficiente sendo necessária diversas leituras somente para atualizar uma pequena informação.

O segundo protótipo seguia a mesma logica, mas a array foi reduzido para uma

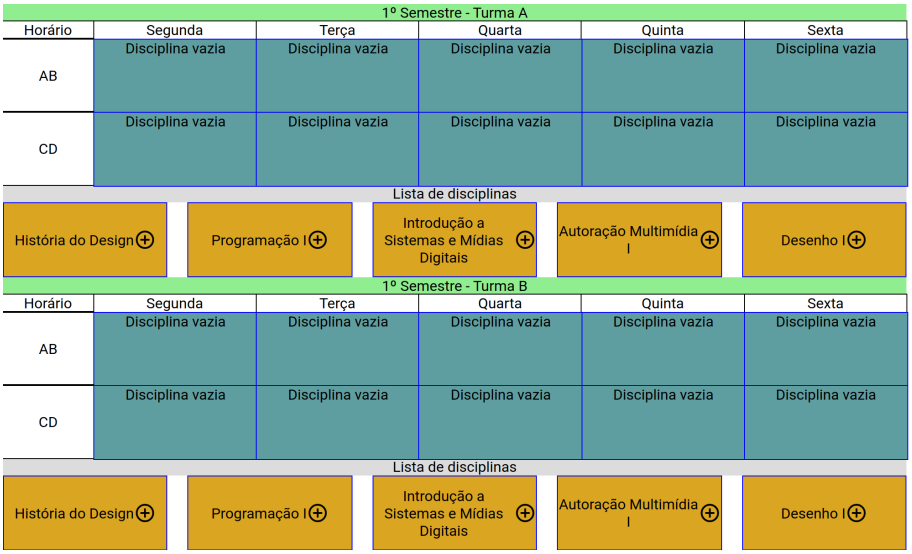
Figura 11 – Print do protótipo com o sistema de tabela inicial



Fonte: elaborado pelo autor (2022)

bidimensional. A array de semestre foi removida e essa informação foi passada para a array de turmas, agora cada turma tem o valor do semestre guardado como pode ser visto na Figura 11.

Figura 12 – Print do protótipo com os sistemas da tabela principal feitos



Fonte: elaborado pelo autor (2022)

4.5.6 Otimizações e melhorias

Com o sistema funcionando conforme foi apresentado na Figura 12 estava na hora de adicionar as outras funções essenciais, no caso as tabelas auxiliares onde as informações geradas pela tabela interativa seriam montadas com o foco em um aspecto, mapa de salas, horário de professores e disciplinas de professores. No entanto, a forma que as informações geradas pela tabela interativa estavam sendo guardada era problemático. Era útil para criar a tabela, mas complicado de extrair as informações necessárias.

4.5.6.1 Sistema de cartões

O sistema de cartão foi a solução para tal problema. Os cartões são objetos que armazenam informações sobre das disciplinas que pertencem a uma turma, no entanto, já que o sistema automaticamente atribui as disciplinas as suas respectivas turmas eles são criados imediatamente assim que a tabela principal é gerada. No entanto, os cartões não são armazenados imediatamente e sim somente quando é adicionado ou alterado um de seus dados, por exemplo, ao ser adicionado um professor, um novo horário ou sala. Por serem objetos possuem padrões flexíveis e escalonáveis podendo receber novas variáveis sem interfira no resto do sistema. Com um sistema de armazenamento mais simples de ser lido tornasse mais prático capturar informações específicas e realizar uma análise de dados.

4.5.6.2 Tabelas de análise de dados

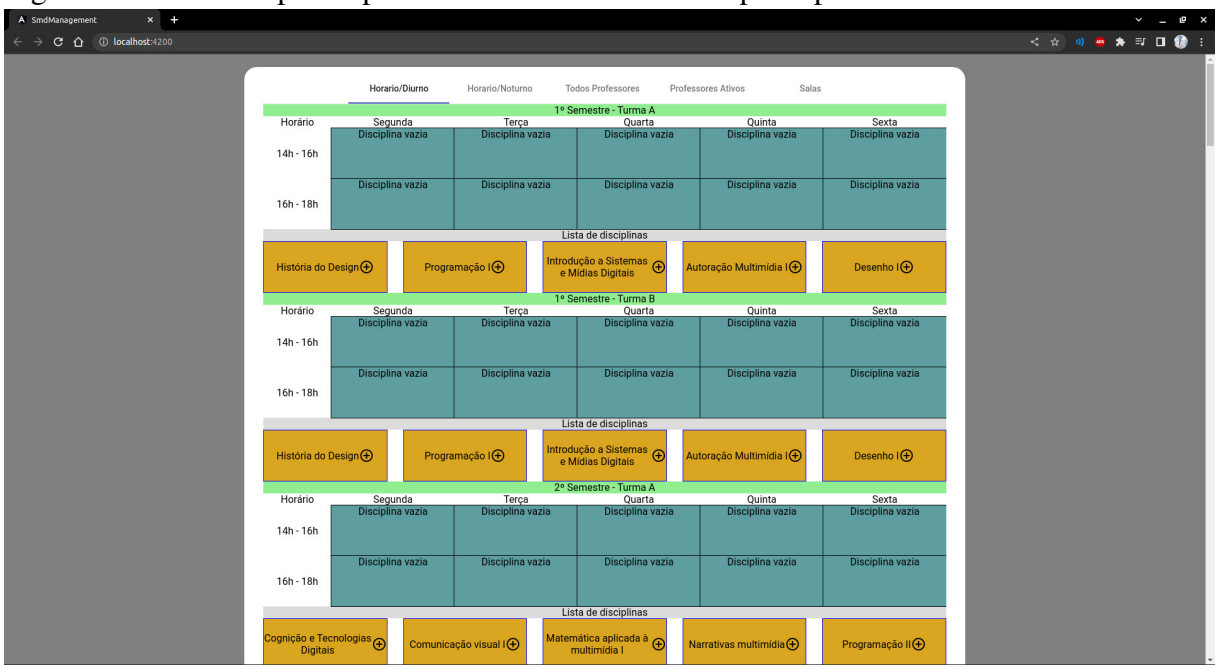
Com dados mais simples de serem coletados, criar as tabelas de análise de dados se tornou menos complexo e mais eficiente. Para quais dados deveriam ser analisados inicialmente foi decido replicar as tabelas de análise já em uso originalmente pela coordenação do curso Sistemas e Mídias Digitais que podem ser observadas nas Figuras 6 e 7. O motivo seria primeiramente para demostrar a capacidade da aplicação como uma melhoria ao método originalmente usado e, além disso entre as possíveis análises de dado a serem feitas essas foram consideradas as mais importantes. O resultado pode ser observado na Figura 13 na qual é possui abas, cada uma representando uma tabela diferente. Uma das grandes vantagens dessas tabelas é a possibilidade de identificar conflitos entre os dados dos cartões.

4.5.6.3 Conflitos

Na tabela principal é possível gerar conflitos entre os dados, por exemplo, caso um professor tenha múltiplas aulas no mesmo horário ou uma sala tenha múltiplas aulas no mesmo horário. A decisão de permitir tais conflitos ocorrerem é deliberada pois, caso fosse feito o bloqueio de conflitos se tornaria tecnicamente impossível fazer qualquer mudança em uma tabela completamente preenchida. Seria necessário fazer múltiplas mudanças não somente na turma da disciplina como em outras turmas somente para mudar alguns horários caso o sistema bloqueasse conflitos. No entanto, o usuário ainda precisa saber que eles estão ocorrendo.

A solução foi mostrar os conflitos somente de forma visual, a forma escolhida foi

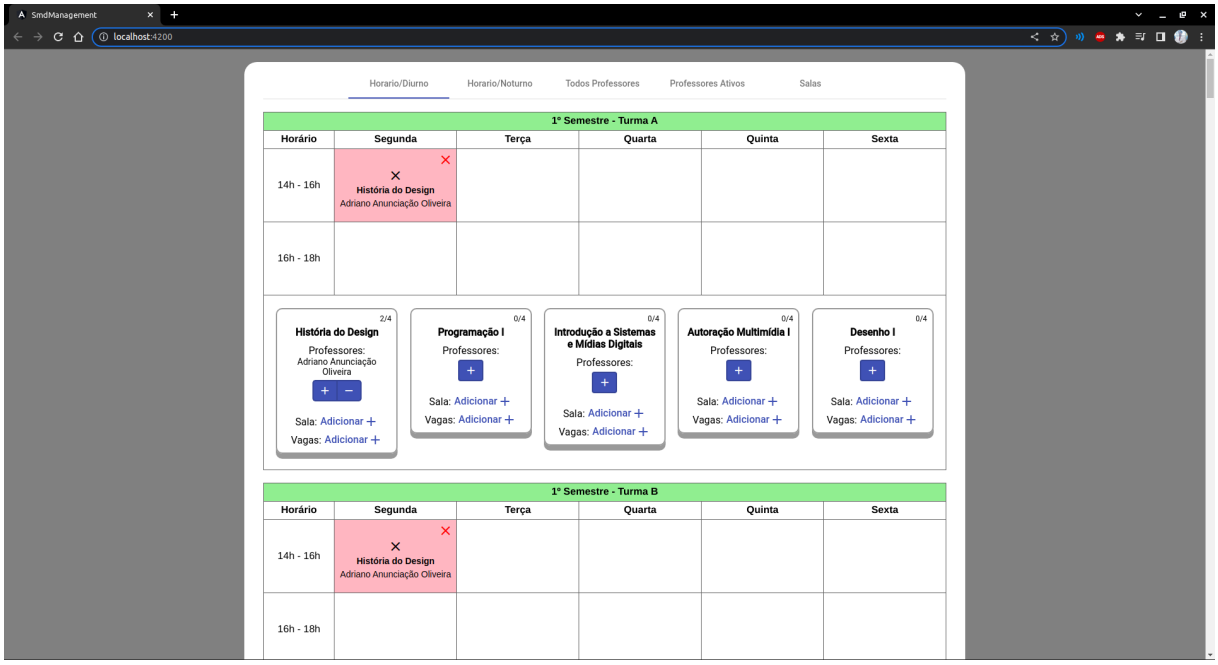
Figura 13 – Print do protótipo com os sistemas da tabela principal feitos



Fonte: elaborado pelo autor (2022)

alterar a cor do campo onde um conflito estiver ocorrendo e usar ícones para demonstrar a quantidade. Como pode ser visto na Figura 14.

Figura 14 – Print do protótipo com os sistemas da tabela principal feitos

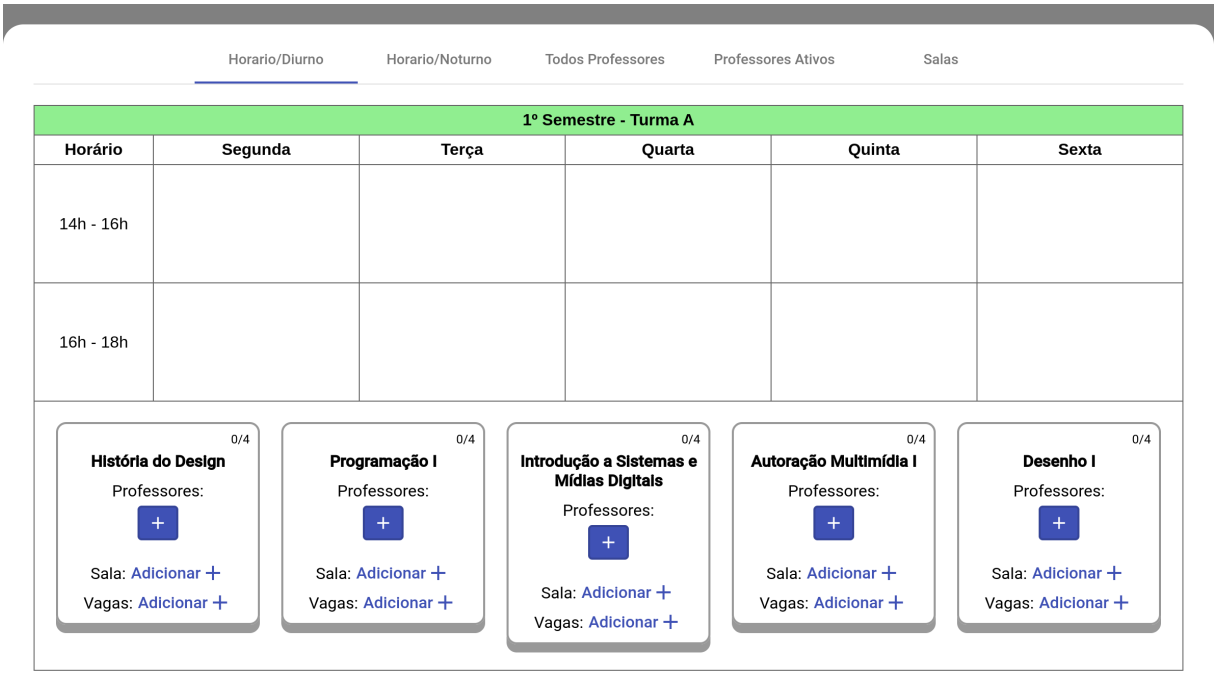


Fonte: elaborado pelo autor (2022)

4.5.7 MVP

Assim resume o desenvolvimento do Oferta Fácil. A aplicação é composta por múltiplas tabelas cada uma designada a uma aba, a primeira tabela e também a principal é totalmente interativa através do componente do angular material *drag-drop* permitindo mudar a posição dos campos da tabela. A baixo da tabela está os controles responsáveis por adicionar ou remover dados como professores, salas e limite de alunos das disciplinas como pode ser visto na Figura 15. Como a quantidade de professores que podem ser adicionados não possui um limite, a forma de adicionar é visualmente e funcionalmente diferente da forma de adicionar uma sala ou quantidade de vagas. Ao realizar qualquer mudança nessa tabela são chamadas as funções presentes no TableService um serviço do angular, ao terminar a execução o sistema de cartões é atualizado salvando a alteração e o array da tabela é atualizado com as novas tabelas.

Figura 15 – Print do mvp do protótipo

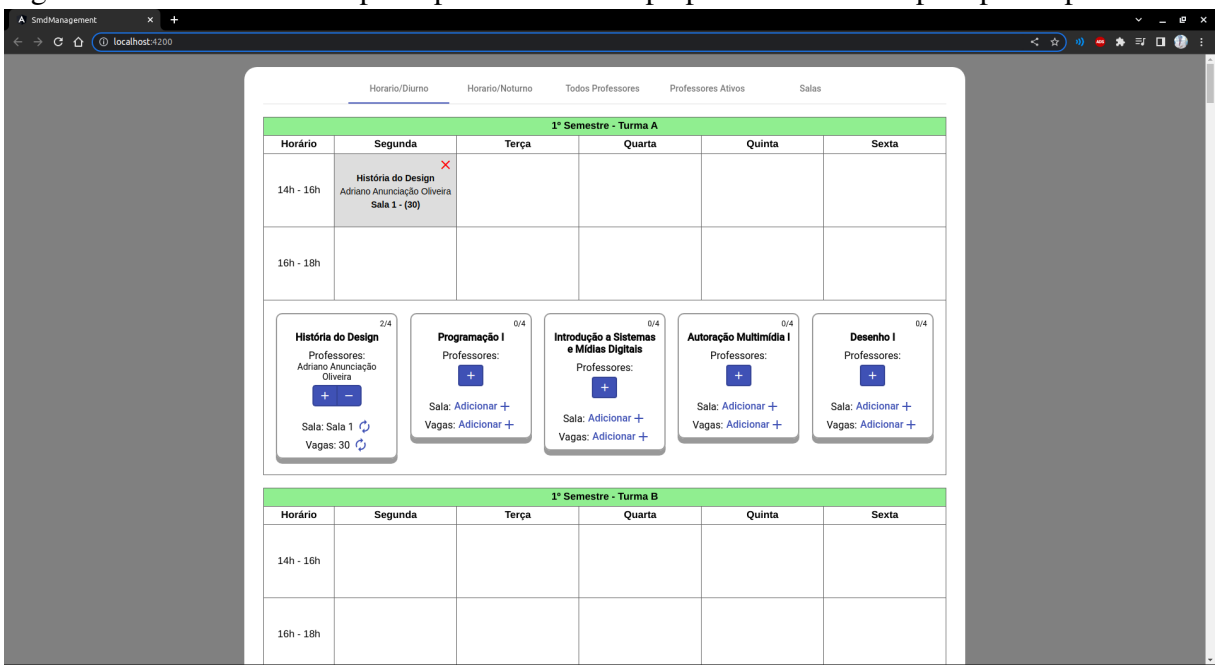


Fonte: elaborado pelo autor(2022)

Ao concluir a construção das tabelas o array é publicado no subject este por vez irá enviar a variável em todas as páginas inscritas nele, assim ao receber a array a view da página irá ser atualizada como pode ser visto na Figura 16. Porém, não será somente atualizada a tabela principal, mas também as tabelas de análise de dados.

A primeira tabela de análise é a tabela encontrada na aba "Todos Professores", como pode ser observado na Figura 16 ela já está preenchida sem a necessidade de intervenção

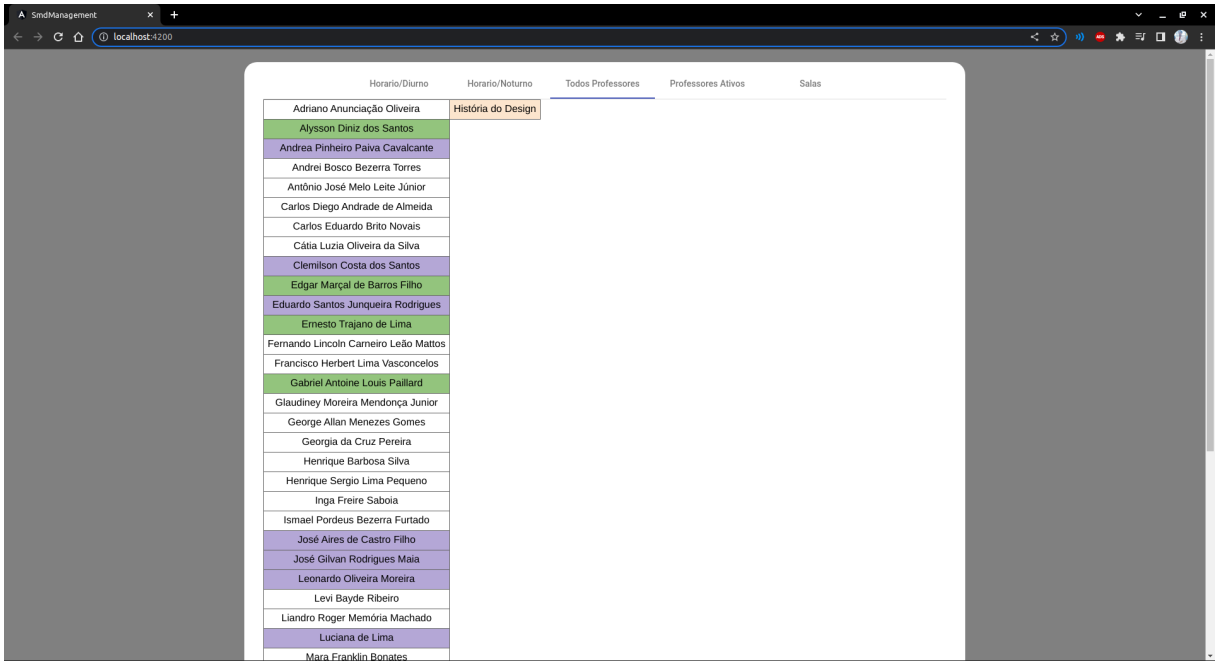
Figura 16 – Print da tabela principal com um campo preenchido do mvp do protótipo



Fonte: elaborado pelo autor(2022)

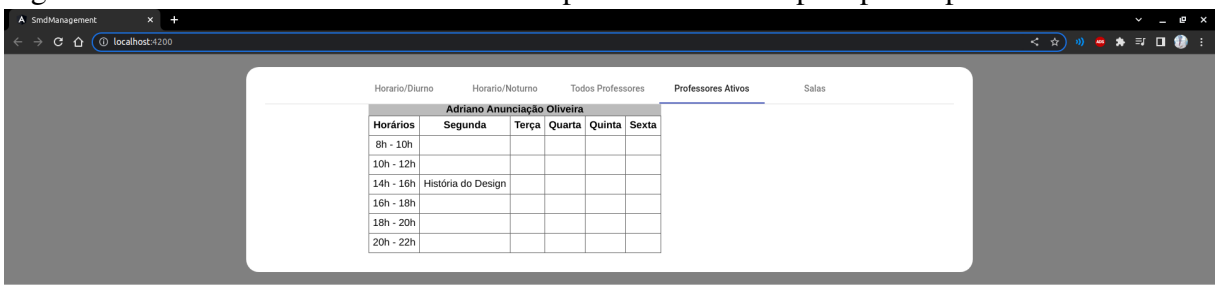
manual. Ela similarmente a original é usada pela coordenação do curso quer extrair a quantidade de disciplinas sendo realizadas por cada professor. A segunda tabela é a localizada na aba Professores ativos, como pode ser observado na Figura 16 ela também já foi preenchida sem a necessidade de intervenção. Também similarmente a original é usada pela coordenação do curso para extrair a distribuição dos horários dos professores.

Figura 17 – Print da tabela de todos os professores do mvp do protótipo



Fonte: elaborado pelo autor(2022)

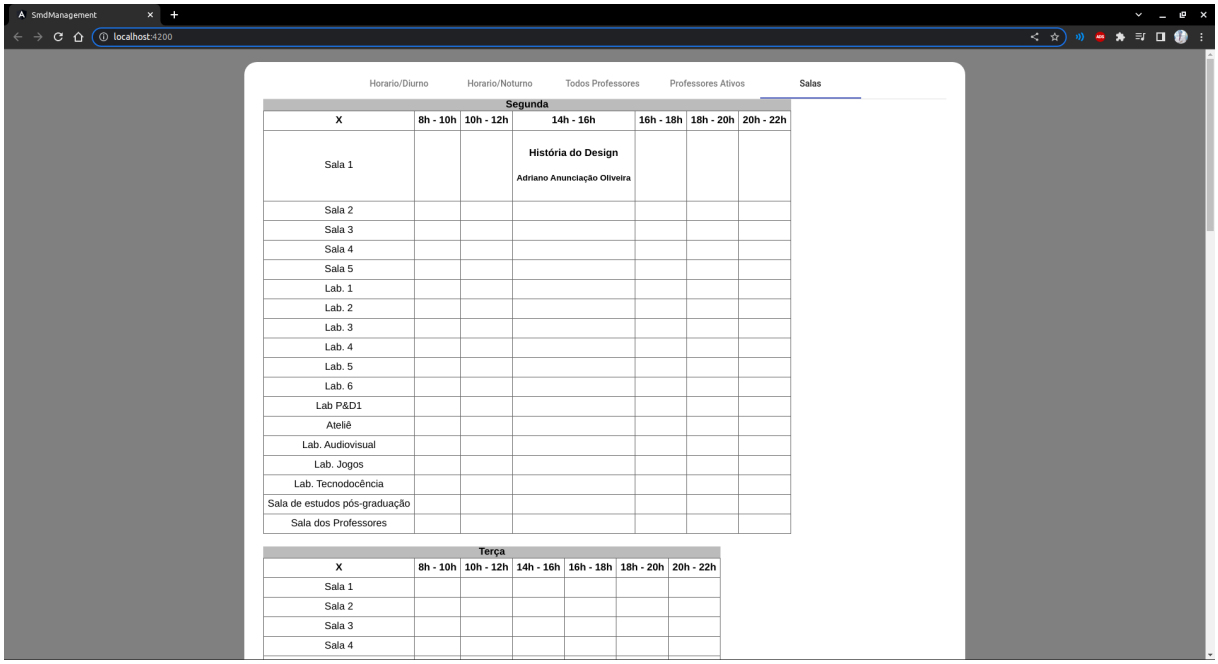
Figura 18 – Print da tabela de horários dos professores do mvp do protótipo



Fonte: elaborado pelo autor(2022)

Por fim a última aba, a aba de salas, representa o mapa de salas e define quais disciplinas e em quais horários ocuparam especifica sala. Sua visualização pode ser vista na Figura 19.

Figura 19 – Print do mapa de salas do mvp do protótipo



Fonte: elaborado pelo autor(2022)

5 CONCLUSÕES E TRABALHOS FUTUROS

Este trabalho relatou o processo de elaboração e desenvolvimento da ferramenta que teria como objetivo auxiliar os coordenadores do curso Sistemas e Mídias Digitais no problema de criação da oferta de turma semestral. Se utilizando dos conceitos e métodos do design thinking foi feito o levantamento dos requisitos necessários para a resolução do problema através de entrevistas. Com base nesses requisitos e em levantamentos bibliográficos sobre o problema de Timetabling, foi proposto a aplicação web Oferta Fácil.

Durante o processo de desenvolvimento foi utilizado os conceitos dos métodos ágeis, em especifico o framework Scrum para gerenciar o desenvolvimento do projeto utilizando a ferramento github projects. Além disso foi feito a descrição do processo de criação e evolução da aplicação desde o primeiro protótipo até o mvp atual, além das tecnologias usadas. Descrevendo as etapas pelos quais cada elemento da aplicação passou e os motivos que os levaram ou a serem descartados ou modificados.

Como trabalhos futuros, pretende-se realizar os seguintes grupos de tarefas: (1) Finalizar o protótipo da aplicação; (2) permitir que os administrador possam inserir os dados essências a aplicação como disciplinas, professores, salas e turmas; (3) adicionar elementos de automação para reduzir tarefas repetitivas; (4) adicionar ajustes automáticos para realizar alterações e correções em um grupo de elementos simultaneamente; (5) escalar as tabelas secundários permitindo que os usuários escolham quais dados desejam ser visualizados; (6) inserir restrições para professores, salas e disciplinas como peculiaridades de salas, bloquear certo horários de professores e quais disciplinas cada professor pode ministrar; (7) permitir a aplicação exportar e ler arquivos JSON contendo toda a oferta de turmas assim tornando possível passar sua oferta para outros usuários; (8) possível integração com o SIGAA ou no SIGAA; (9) possível servidor dedicado permitindo salvar em um banco de dados as alterações realizadas e associá-las a um login; (10) realizar a validação da aplicação e suas funcionalidades com teste de usuários de diferentes perfis e características;

REFERÊNCIAS

AMBROSE, G.; HARRIS, P. **Design thinking: Coleção design básico**. [S.l.]: Bookman Editora, 2016. ISBN 978-2-940411-17-7.

BARDADYM, V. A. Computer-aided school and university timetabling: The new wave. **International conference on the practice and theory of automated timetabling**, p. 22–45, 8 1995.

CRUZ, F. **Scrum e PMBOK unidos no Gerenciamento de Projetos**. [S.l.]: Brasport, 2013. ISBN 978-85-7462-594-5.

KHAN, M.; KHAN, S. S. Data and information visualization methods, and interactive mechanisms: A survey. **International Journal of Computer Applications**, v. 34, n. 1, p. 1–14, 2011.

KRISTIANSEN, S.; STIDSEN, T. R. A comprehensive study of educational timetabling—a survey. **DTU Management Engineering. DTU Management Engineering Report**, n. 8, p. 22–45, 2013.

MARIANO, C. L. **Benchmarking javascript frameworks.** Dissertação (Mestrado) — Technological University Dublin, 1 2017.

MCCOLLUM, B. University timetabling: Bridging the gap between research and practice. **E Burke, HR, ed.: PATAT**, p. 15–35, 2006.

PRIKLADNICKI, R.; WILLI, R.; MILANI, F. **Métodos ágeis para desenvolvimento de software**. [S.l.]: Bookman Editora, 2014. ISBN 978-85-8260-208-9.

SADALAGE, P. J.; FOWLER, M. **NoSQL essencial: um guia conciso para o mundo emergente da persistência poliglota**. [S.l.]: Novatec Editora, 2019.

SCHAERF, A. A survey of automated timetabling. Kluwer Academic Publishers, v. 13, n. 2, p. 87–127, 1999.

VALENTE, M. T. **Engenharia de Software Moderna: Princípios e Práticas para Desenvolvimento de Software com Produtividade**. [S.l.]: Bookman Editora, 2020. v. 1. ISBN 978-65-00-00077-1.

WREN, A. Scheduling, timetabling and rostering—a special relationship? **Management Science**, p. 46–75, 1996.

APÊNDICE A – REQUISITOS FUNCIONAIS

- RF001 - Visualizar as tabelas: o usuário deve ser capaz de visualizar as diferentes tabelas geradas pelo sistema;
- RF002 - Trocar de tabela: o usuário pode selecionar para visualizar as diferentes tabelas disponíveis;
- RF003 - CRUD de disciplina: o administrador pode cadastrar, consultar, alterar ou remover disciplinas da aplicação.
- RF004 - CRUD de professor: o administrador pode cadastrar, consultar, alterar ou remover professores da aplicação.
- RF005 - CRUD de sala: o administrador pode cadastrar, consultar, alterar ou remover salas da aplicação.
- RF006 - CRUD de turma: o administrador pode cadastrar, consultar, alterar ou remover turmas da aplicação.
- RF007 - Adicionar disciplina a uma turma: o usuário pode selecionar entre as disciplinas disponíveis e adicioná-la a uma turma;
- RF008 - Adicionar professores a uma disciplina: o usuário pode selecionar entre os professores disponíveis e adicioná-lo a disciplina;
- RF009 - Adicionar sala a disciplina: o usuário pode selecionar entre as salas disponíveis e adicioná-lo a disciplina;
- RF010 - alterar quantidade de vagas da disciplina: o usuário pode alterar a quantidade de vagas disponíveis da disciplina;
- RF011 - Alterar horário de uma disciplina: o usuário pode mudar o horário da disciplina;
- RF012 - Diferenciar disciplinas pela cor: o usuário deve ser diferenciar as diferentes categorias das disciplinas pelas cores usadas;
- RF013 - Ver conflitos: o usuário deve ser capaz de visualizar os diferentes conflitos acontecendo em cada disciplina, sala ou professor;

APÊNDICE B – ESPECIFICAÇÃO DE REQUISITOS DO SISTEMA

RF001 A aplicação deverá possuir múltiplas tabelas, cada uma focado em um certo tipo de informação, o sistema tem que ser capaz de analisar os dados gerados pela tabela principal e gerar as outras tabelas sem atraso.

RF002 A aplicação deverá possuir múltiplas tabelas, cada uma focado em um certo tipo de informação, o usuário tem que ser capaz de mudar qual tabela ele está visualizando no momento de preferência com o menor número de cliques possível.

RF003/RF004/RF005/RF006 O sistema deve ser capaz criar, alterar, mostrar e excluir os quatro principais dados do sistema sendo eles as listas de disciplinas, professores, salas, turmas. Além disso, o administrador deve ser capaz de realizar tais ações a qualquer momento.

RF007 Durante a criação da oferta de turmas o usuário deve ser capaz de associar uma disciplina a uma turma através de uma lista na qual contem todas as disciplinas disponíveis. Quando a disciplina associação estiver completa a disciplina poderá ser associada com outros dados como sala, professor e horário.

RF008 Durante a criação da oferta de turmas o usuário deve ser capaz de associar um professor a uma disciplina que já foi associada a uma turma através de uma lista na qual contem todas os professores disponíveis.

RF009 Durante a criação da oferta de turmas o usuário deve ser capaz de associar uma sala a uma disciplina que já foi associada a uma turma através de uma lista na qual contem todas as salas disponíveis.

RF010 Durante a criação da oferta de turmas o usuário deve ser capaz alterar a quantidade de vagas de uma disciplina que já foi associada a uma turma.

RF011 Durante a criação da oferta de turmas o usuário deve ser capaz de alterar e adicionar horários a uma disciplina que já foi associada a uma turma através de uma lista na qual contem todos os horários disponíveis.

RF012 As diferentes disciplinas que foram associadas a uma turma devem conter cores diferentes baseadas nas suas características.

RF013 Ao realizar as associações, conflitos de horários e salas irão ocorrer assim o sistema deve alertar o usuário que esteja fazendo as associações que ocorreu um conflito.

APÊNDICE C – REGRAS DE NEGÓCIO

- RN001 - Horários só podem ser adicionados a uma aula caso o limite de horas da disciplina associado a aula não tenha sido ultrapassado.
- RN002 - Horários das aulas podem se alterados seguindo os limites de períodos e dias fornecidos pelo sistema.
- RN003 - Ao alterar um horário de uma aula, caso o novo horário não possua nenhuma aula ocorrerá uma troca de horários, porém caso o novo horário já possua uma aula ocorrerá uma permutação de horário entre as aulas.
- RN004 - Salas somente podem ser adicionadas a uma aula caso a aula não possua uma sala já associada devido ao limite de uma sala por aula.
- RN005 - Caso uma aula já possua uma sala associada, o sistema deve indicar que agora é possível trocar a sala e não adicioná-la.
- RN006 - Uma quantidade de vagas somente podem ser adicionadas a uma aula caso a aula não possua uma quantidade de vagas já associada.
- RN007 - Caso uma aula já possua uma quantidade de vagas associada, o sistema deve indicar que agora é possível trocar a quantidade de vagas e não adicioná-la.
- RN008 - Qualquer professor pode ser adicionado a uma aula, desde que ele já não tenha sido associado a aula.
- RN009 - Qualquer professor pode ser removido de uma aula, desde que ele esteja associado a aula.
- RN010 - Aulas não possuem um limite de quantos professores podem ser associados.
- RN011 - Todas as alterações sofridas nas aulas devem ser imediatamente salvas no banco de dados.
- RN012 - Toda alteração realizada no banco de dados deve ser processada em tabelas de analise de dados.
- RN013 - Toda alteração realizada no banco de dados deve ser processada para buscar e identificar conflitos nas aulas.
- RN014 - Caso um conflito seja identificado, deve-se mostrar visualmente quais aulas possuem dados conflitantes.

APÊNDICE D – HISTÓRIAS ENCONTRADAS NO BACKLOG

1. Como organizador, eu quero poder ver todas as disciplinas do meu curso para eu poder organizá-las;
2. Como organizador, eu quero poder adicionar uma disciplina a tabela para que eu inicie a criação da oferta de turmas;
3. Como organizador, eu quero associar professores a uma disciplina para que eu inicie a criação da oferta de turmas;
4. Como organizador, eu quero poder trocar a posição das disciplinas na tabela de forma fácil para que eu organize a oferta de turmas;
5. Como organizador, eu quero expandir uma disciplina de 2h para uma de 4h para eu compreender melhor a oferta de turmas;
6. Como organizador, eu quero poder permutar duas disciplinas de forma fácil para eu organizar eficientemente a oferta de turmas;
7. Como organizador, eu quero que as disciplinas tenham cores baseado nas suas unidade curriculares para eu compreender melhor a oferta de turmas;
8. Como organizador, eu quero que as disciplinas estejam categorizadas baseado em seus tipos para eu compreender melhor a oferta de turmas;
9. Como organizador, eu quero que as disciplinas possam ser pareadas para que facilite a organização da oferta de turma;
10. Como organizador, eu quero saber se uma disciplina já esta com sua cota de horas completa para que eu organize a oferta de turmas;
11. Como organizador, eu quero saber se uma disciplina já possui um professor associado para que eu organize a oferta de turmas;
12. Como organizador, eu quero saber se uma disciplina foi adicionada a tabela principal para que eu organize a oferta de turmas;
13. Como organizador, eu quero que a tabela principal seja fácil de visualizar para eu compreender melhor a oferta de turmas;
14. Como organizador, eu quero que a tabela de professores seja fácil de visualizar para eu compreender melhor a oferta de turmas;
15. Como organizador, eu quero que a tabela de professores esteja de acordo com a tabela de horários para que facilite a organização da oferta de turma;
16. Como organizador, eu quero que a tabela de horário de professores seja fácil de visualizar

para eu compreender melhor a oferta de turmas;

17. Como organizador, eu quero que a tabela de horário de professores esteja de acordo com a tabela de horários para que facilite a organização da oferta de turma;
18. Como organizador, eu quero ter múltiplas versões do conjunto de tabelas para eu poder ter diversas escolhas;
19. Como organizador, eu quero trocar entre de cada conjunto de tabelas para eu poder ver todas as minhas opções;
20. Como organizador, eu quero trocar entre as diferentes tabelas para eu compreender melhor a oferta de turmas;
21. Como organizador, eu quero ver antigas ofertas de turma para que eu compreenda melhor a divisão das disciplinas;
22. Como organizador, eu quero exportar relatórios de dados combinados de todas ou algumas ofertas de turma para que eu compreenda melhor a divisão das disciplinas;
23. Como organizador, eu quero compartilhar minha oferta de turma para outros professores para nós podermos debater a oferta;
24. Como organizador, eu quero saber quantas horas cada professor está nesta oferta de turmas para que eu organize a oferta de turmas;
25. Como organizador, eu quero saber se um professor está com conflito nos horários para que eu organize a oferta de turmas;