**CENTRO PAULA SOUZA**

**ESCOLA TÉCNICA ESTADUAL DE TABOÃO DA SERRA**

**TÉCNICO EM INFORMÁTICA**

**ELMER CAMPOS DE ARAÚJO**

**MATTEO MARTINS CAMARGO**

**RAPHAEL SIQUEIRA GABIA**

**VINICIUS PEREIRA FLORIANO**

**VINICIUS STEPHANTO OLIVEIRA**

**VITOR SILVA CAVALCANTE**

**SISTEMA DE GERENCIAMENTO DE GRADE HORÁRIA**

**Projeto Klass**

**Taboão Da Serra**

**2020**

**ELMER CAMPOS DE ARAÚJO**

**MATTEO MARTINS CAMARGO**

**RAPHAEL SIQUEIRA GABIA**

**VINICIUS PEREIRA FLORIANO**

**VINICIUS STEPHANTO OLIVEIRA**

**VITOR SILVA CAVALCANTE**

**SISTEMA DE GERENCIAMENTO DE GRADE HORÁRIA**

**Projeto Klass**

Trabalho de Conclusão de Curso apresentado ao Curso Técnico em Informática da Escola Técnica Estadual de Taboão Da Serra, como requisito parcial para obtenção do título de Técnico de Informática.

Orientadores: Prof. Marcos Renato Alves

Prof. Fabiano Jorge Oliveira

**Taboão Da Serra**

**2020**

**FOLHA DE APROVAÇÃO**

**ELMER CAMPOS DE ARAÚJO**

**MATTEO MARTINS CAMARGO**

**RAPHAEL SIQUEIRA GABIA**

**VINICIUS PEREIRA FLORIANO**

**VINICIUS STEPHANTO OLIVEIRA**

**VITOR SILVA CAVALCANTE**

**SISTEMA DE GERENCIAMENTO DE GRADE HORÁRIA**

**Projeto Klass**

Trabalho de conclusão de curso apresentado como requisito parcial para obtenção do título de Técnico de Informática em 11/12/2020, pela Escola Técnica Estadual De Taboão Da Serra.

Aprovado em: XX de dezembro de 2020.

**Banca Examinadora**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Nome do orientador, sua titulação e Instituição a que pertence).

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(nome, titulação e instituição a que pertence).

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(nome, titulação e instituição a que pertence).

**RESUMO**

O trabalho presente tem como objetivo apresentar a ideia inicial do aplicativo "Klass". Teremos duas formas de se comunicar com nossos clientes. Um aplicativo web e um aplicativo mobile. O projeto “Klass” abranje duas esferas bem correlacionadas, a primeira de criação de grades horárias e a segunda é de tornar o acesso a esse documento de uma forma mais facilitada e eficiente, no caso, o nosso foco no momento. Nosso cliente principal foi a Faculdade Lusófona de São Paulo que nos passou a demanda para ter um trabalho mais simples e eficiente, nos foi passado a lógica atual em que se é usada para realização dos horários.

**Palavras-chave:** Web. Desktop. Grades Escolares. Klass.

**ABSTRACT**

The present work aims to present the initial idea of ​​the "Klass" application. We will have two ways to communicate with our customers. A web / desktop application and a mobile application. The “Klass” project encompasses two well-correlated spheres, the first of which creates hourly schedules and the second is to make access to this document easier and more efficient, in this case, our focus at the moment. Our main client was Faculdade Lusófona de São Paulo, which gave us the demand to have a simpler and more efficient job.

**Key-words:** Klass. School schedules. Web. Desktop.

**LISTAS DE FIGURAS**

[Figura 1 - Diagrama de sequências 13](#_Toc55653927)

[Figura 2 - Diagrama de classe 14](#_Toc55653928)

[Figura 3 - Diagrama de casos de uso 14](#_Toc55653929)

[Figura 4 - Diagrama de atividades 15](#_Toc55653930)

[Figura 5 - Diagrama Entidade-Relacionamento (DER) 15](#_Toc55653931)

[Figura 6 - Tela de Cadastro 25](#_Toc55653932)

[Figura 7 - Tela de Login 26](#_Toc55653933)

[Figura 8 – Tela Escolha de Tema 27](#_Toc55653934)

[Figura 9 - Tela e-mail de ativação 28](#_Toc55653935)

[Figura 10 - Tela Home 29](#_Toc55653936)

[Figura 11 - Tela de Cronograma 29](#_Toc55653937)

[Figura 12 - Tela criação nova escola 30](#_Toc55653938)

[Figura 13 - Tela de nome, descrição e de tipo de escola 32](#_Toc55653939)

[Figura 14 - Tela escolha de ícone para a escola 32](#_Toc55653940)

[Figura 15 - Tela criação turnos e horários 33](#_Toc55653941)

[Figura 16 - Tela criação de cursos e módulos 35](#_Toc55653942)

[Figura 17 - Tela criação de Matérias e Conexão aos módulos 36](#_Toc55653943)

[Figura 18 - Criação Professores e Matérias 37](#_Toc55653944)

[Figura 19 - Tela de atribuição turnos aos módulos 38](#_Toc55653945)

[Figura 20 - Tela de dashboard 39](#_Toc55653946)

[Figura 21 - Tela erro na página 40](#_Toc55653947)

[Figura 22 - Tela participação de uma escola 40](#_Toc55653948)

[Figura 23 - Tela primeiro acesso 41](#_Toc55653949)

[Figura 24 - Tela de vídeo tutorial 42](#_Toc55653950)

[Figura 25 - Tela de perfil 43](#_Toc55653951)

[Figura 26 - Tela de turmas 44](#_Toc55653952)

[Figura 27 - Telas de cursos 45](#_Toc55653953)

[Figura 28 - Tela de turnos 46](#_Toc55653954)

[Figura 29 - Tela de módulos 47](#_Toc55653955)

[Figura 30 - Tela de docentes 48](#_Toc55653956)

[Figura 31 - Tela de períodos 49](#_Toc55653957)

[Figura 32 - Tela de relatório 50](#_Toc55653958)

[Figura 33 - Tela para participar de uma escola 51](#_Toc55653959)

**LISTAS DE GRÁFICOS**

**LISTAS DE TABELAS**

[Tabela 1 - Cronograma Etapa 1 22](#_Toc55653996)

[Tabela 2 - Cronograma Etapa 2 23](#_Toc55653997)

[Tabela 3 - Cronograma Etapa 3 23](#_Toc55653998)

[Tabela 4 - Cronograma Etapa 4 24](#_Toc55653999)

**Sumário**

[1. INTRODUÇÃO 7](#_Toc56805543)

[2. PROBLEMA DE PESQUISA 7](#_Toc56805544)

[3. JUSTIFICATIVA 8](#_Toc56805545)

[4. METODOLOGIA 8](#_Toc56805546)

[4.1. HIPÓTESE 9](#_Toc56805547)

[5. OBJETIVOS 9](#_Toc56805548)

[5.1. OBJETIVOS GERAIS 9](#_Toc56805549)

[5.2. OBEJTIVOS ESPECÍFICOS 10](#_Toc56805550)

[6. FATOR HUMANO NA GERÊNCIA ACADÊMICA 10](#_Toc56805551)

[7. LEVANTAMENTO DE REQUISITOS 11](#_Toc56805552)

[7.1. REQUISITOS FUNCIONAIS 11](#_Toc56805553)

[7.2. REQUISITOS NÃO FUNCIONAIS 12](#_Toc56805554)

[7.3. EVOLUÇÃO DO SISTEMA 13](#_Toc56805555)

[7.4. DIAGRAMAS UML 13](#_Toc56805556)

[7.4.1. DIAGRAMA DE SÊQUENCIAS 13](#_Toc56805557)

[7.4.2. DIAGRAMA DE CLASSES 14](#_Toc56805558)

[7.4.3. DIAGRAMA DE CASOS DE USO 14](#_Toc56805559)

[7.4.4. DIAGRAMAS DE ATIVIDADES 15](#_Toc56805560)

[7.4.5. DIAGRAMA ENTIDADE-RELACIONAMENTO (DER) 15](#_Toc56805561)

[8. FUNCIONALIDADES 16](#_Toc56805562)

[8.1. AUTENTICAÇÃO 16](#_Toc56805563)

[8.2. PRIMEIRO ACESSO 16](#_Toc56805564)

[8.3. DASHBOARD 16](#_Toc56805565)

[8.4. CADASTRO 17](#_Toc56805566)

[8.5. RELATÓRIO 18](#_Toc56805567)

[8.6. PÓS RELATÓRIO 18](#_Toc56805568)

[8.7. CONEXÕES 18](#_Toc56805569)

[8.8. CONFIGURAÇÃO 18](#_Toc56805570)

[8.9. ADICIONAIS 18](#_Toc56805571)

[9. ESPECIFICAÇÕES DO PROJETO 19](#_Toc56805572)

[9.1 ESCOPO DO PROJETO 19](#_Toc56805573)

[9.2. LIMITAÇÕES DO PROJETO 22](#_Toc56805574)

[11. TELAS PRINCIPAIS 22](#_Toc56805575)

[CRONOGRAMA 50](#_Toc56805576)

[CONSIDERAÇÕES FINAIS 50](#_Toc56805577)

[REFERÊNCIAS 51](#_Toc56805578)

# 1. INTRODUÇÃO

Vemos atualmente, cada vez mais, seres humanos com menos tempo e mais atividades a realizar, principalmente em uma sociedade globalizada como a atual, sistemas e programas através do avanço tecnológico da informação que facilitem a vida das pessoas, acabam se tornando cada vez mais necessárias. Assim sendo e pensando no funcionamento das escolas por meio vida dos coordenadores, o projeto Klass através do diretor da Faculdade Lusófona de São Paulo, Cesar Cavinato, que nos trouxe a demanda de um sistema que se automatiza o processo de criação de grade horárias de aulas o que manualmente demanda um exaustivo trabalho e muito tempo.

O trabalho presente tem como objetivo apresentar a o funcionamento do aplicativo Klass. O projeto Klass abrange duas esferas bem correlacionadas, a primeira a facilitação na criação de grades horárias e a segunda é de tornar o acesso a esse documento de uma forma mais simples e eficiente.

O aplicativo traz, a facilitação da vida profissional dos professores ou coordenadores que irão usufruir do aplicativo, a facilitação de seu trabalho ocorrerá principalmente pela otimização de seu tempo. Levando a gastar o mesmo em tarefas mais oportunas como a qualidade da vida acadêmica de seus alunos.

# 2. PROBLEMA DE PESQUISA

No âmbito de nossas pesquisas, pudemos observar que, atualmente, há um grande problema em diversas instituições de ensino, a criação de grades horárias e a acessibilidade a elas. Este problema se torna visível quando percebemos que em nenhuma das diversas escolas que tivemos contato os horários são feitos de forma automática com auxílio de alguma tecnologia, ou seja, são feitos manualmente.

Até mesmo em faculdades, onde existe uma demanda de diversos professores e salas, todo esse quebra-cabeça fica por conta de uma única pessoa ou pequeno grupo de pessoas. Após o problema ser confirmado pelas instituições que foram entrevistadas, notamos que existem sites e aplicativos onde é possível montar grades de horários, mas, não são simples e intuitivos, isso faz com que o problema que poderia ser resolvido de uma forma rápida, acaba sendo mais difícil do que deveria.

O grande estresse do cotidiano dos docentes acadêmicos resulta em uma banalização do problema, em foco os coordenadores que esteja submetido a essa situação, causando um desinteresse em grupos e equipes de programação em resolver esse problema, gerando uma ociosidade de referências sobre o assunto.

# 3. JUSTIFICATIVA

A falta de oferta no mercado por esse produto, viabilizou a elaboração de um projeto de pesquisa e entrevistas a respeito do tema e de sua demanda. Essa deficiência no sistema educacional motivou o projeto, sendo importante ressaltar o vínculo pessoal que o grupo possui com o tema, já que ele está inserido em nosso meio acadêmico. O desenvolvimento de um sistema eficiente e capaz de automatizar o método de criação de horários acadêmicos é de suma importância para a coordenação responsável pelo êxito dessa tarefa.

Este trabalho se justifica pela necessidade que todas as instituições acadêmicas têm de criar o horário escolar das aulas. Visto que isso se torna um problema, pois demanda muito tempo, é complicado alocar as aulas respeitando os horários dos professores e algumas vezes o resultado pode ficar insatisfatório.

# 4. METODOLOGIA

Essa é uma das principais deficiências do sistema educacional, pois, todas as instituições que dispõe dessa metodologia de educação não contam com um sistema eficaz para a realização dessa tarefa. Com o resultado de nossas pesquisas vimos que esse processo poderia ser muito mais eficaz caso houvesse o auxílio de um sistema que apenas recebesse essas informações e as interpretassem para montá-lo automaticamente.

Diante desta questão, realizamos pesquisas de caráter exploratório fundamentada em análise de documentos e entrevistas, defronte a isso usamos fontes primárias. Sendo assim, foram feitas algumas entrevistas com coordenadores e secretarias de escolas públicas, como a Etec de Taboão da Serra e particulares como o Colégio Nova Geração, além do diretor da faculdade de Lusófona de São Paulo, com o objetivo de entender essas complicações e dificuldades no processo de organização dos horários acadêmicos, aonde nos apresentaram a lógica e raciocínio na formulação das grades horárias. Os resultados serão medidos qualitativamente, pois deve-se analisar se o sistema está funcional e se está realizando o que é proposto a ele ou se existe alguma irregularidade.

## 4.1. HIPÓTESE

Com o coordenador de uma instituição, antes do início definitivo do ano letivo, vá organizar a grade horária de sua instituição, que conta com muitos professores, alunos, turmas e matérias, ele terá uma grande trabalho pela frente, que será cansativo e ocupará muito de seu tempo, até de fato conseguir uma possível solução para seus horários. Como primeiro trabalho de pesquisa e solução, buscamos atender uma necessidade local e específica, mas que, se bem sucedida a ideia possa abranger novas instituições e assim abranger para oferta no mercado.

Então nosso projeto irá solucionar esses problemas, com a criação de um programa para gerar grades horárias. O objetivo é desenvolver esse projeto para que seja acessível e intuitivo, para que escolas, faculdades ou outras instituições acadêmicas optem por usar nossa ferramenta para desenvolver as grades de horários das aulas. Dessa forma, o software irá criar as grades de horário onde o usuário irá inserir apenas as informações essenciais para que possa ser gerado de forma automática e não crie conflitos de horários.

# 5. OBJETIVOS

## 5.1. OBJETIVOS GERAIS

Desenvolver um aplicativo web e um aplicativo mobile. Trazer para o mercado a oferta de um produto eficiente e tão necessário quanto à sua aplicação. O aplicativo web terá como objetivo a criação da grade horária, enquanto o objetivo de nosso aplicativo mobile será facilitar o acesso e visualização dos horários escolares por alunos e professores de suas respectivas escolas.

## 5.2. OBEJTIVOS ESPECÍFICOS

O projeto Klass, além de desenvolver o horário escolar pelo app web/desktop, ele também nos permite aos usuários tenha acesso a toda sua grade horária acadêmica, sendo assim, os alunos sabem: o dia, a hora, a disciplina e o professor destinado, além do turno correspondente em que devem comparecer na unidade escolar, ou forma de ensino utilizada. Ao passo que os professores consultam a unidade escolar, a turma, a disciplina respectiva, os horários e os dias em que devem lecionar. Tudo isso de uma forma bem acessível e de onde estiver, basta apenas ter a disponibilidade de dispositivo celular.

# 6. FATOR HUMANO NA GERÊNCIA ACADÊMICA

Organizar os horários acadêmicos é tanto uma necessidade quanto um desafio geral de todo o sistema educacional, requisitando um planejamento para atender as diferentes demandas e otimizar o desempenho das atividades. Dada a importância dessa organização, cabe ao corpo docente iniciar o quanto antes esse processo de planejamento com o intuito de esquivar-se de possíveis incompatibilidades que ele pode gerar, o que muitas vezes consiste num trabalho complexo e exaustivo, (Carolina da Costa Santos; Fátima Pereira; Amélia Lopes, 2018) a intensificação do trabalho docente em diferentes países e apontam para a falta de tempo que emerge do excessivo acúmulo de tarefas.

Dessa maneira, apresentamos uma solução que possa mudar esse cenário e facilitar a vida e a saúde de nossos coordenadores, com um sistema que gere uma grade horária sozinho, apenas com as informações depositadas no nosso sistema que serão interpretadas e reorganizadas. Na rede púbica não há oferta desse produto, embora possua uma alta demanda e uma alta complexidade de desenvolvimento.

“Em um processo longo e gradual de mudanças organizacionais, a partir dos anos 1960 a 1980, o serviço público transformou práticas, hábitos, valores e rotinas organizacionais, seja de forma planejada endogenamente, como resultado de medidas adotadas, na maioria das vezes normativamente, a partir dos conhecidos “planos de modernização” (Milton Cordeiro; Maria das Graças; Durbens Martins, 2014, p.72, p.83)

O processo de implantação ocorre gradualmente e se torna complexo para os usuários no primeiro período, mas após a familiarização no uso e elucidação sobre o uso a ferramenta se torna uma forma de mudança não só no esforço individual, mas também no funcionamento da gestão como um todo, (Milton Cordeiro; Maria das Graças; Durbens Martins, 2014) quanto aos usuários do sistema, houve um ganho considerável de tempo, movido pela agilidade em que as informações passaram a ser de fácil acesso e sem limitações para a melhora dos resultados, já que os principais usuários: professores, gestores e estudantes passaram a usar o sistema integralmente.”.

# 7. LEVANTAMENTO DE REQUISITOS

Ao analisarmos o sistema como um todo através de 3 entrevistas com o diretor da Faculdade Lusófona de São, Cesar Cavinato, e entendemos os requisitos funcionais e não funcionais do sistema que serão de suma importância para o desenvolvimento dele. Dessa forma, a lógica do sistema por meio do cliente é que o coordenador tenha maior liberdade e menos trabalho, assim sendo ele cria uma escola, como se fosse o molde do ambiente escolar aonde os professores preenchem um formulário com os seus dados esses que retornam ao coordenador para validação e mediação, que automaticamente gera a grande.

## 7.1. REQUISITOS FUNCIONAIS

* Cadastro de usuário;
* Login do usuário;
* Edição de informações do usuário;
* Cadastro de Escola;
* Cadastro de Turnos;
* Cadastro de Cursos;
* Cadastro de Módulos;
* Cadastro de Matérias;
* Cadastro de Professores;
* Cadastro de Turmas;
* Formulário de Questões para Professores;
* Entrar em uma Escola;
* Gerar relatório;
* Editar relatório;
* Visualizar relatório.

Exemplo:

O portal deve oferecer a busca de passagens aéreas. Informações: Local de origem, local de destino, data de partida, data de volta, quantidade de adultos, quantidade de crianças, idade das crianças. Regras: O sistema deve permitir que o visitante ou o cliente do portal possam executar a busca fornecendo as informações obrigatórias e, caso haja resultado para a consulta, o portal deve disponibilizar as passagens aéreas com seus preços, as companhias disponíveis, escala e conexões (se houver), origem, destino e horário. Caso não haja disponibilidade, deve ser informado ao cliente que não houve resultado para a consulta. A obtenção dos dados de passagens de avião deve ser efetivada através de uma interface de integração entre o portal e as companhias aéreas. • O portal deve validar se as informações sobre local de origem, local de destino, data de partida, data da volta e número de pessoas estão preenchidos. • A data de partida deve ser maior ou igual à data corrente. • A data de volta deve ser maior ou igual à data de partida. • O número de adultos ou o número de crianças deve ser maior que zero. Por padrão, o portal deve preencher o número de adultos com 1(um). • Caso o número de crianças seja maior que 0 (zero), a idade deve ser preenchida. • Caso alguma das informações acima não cumpra essas regras, o portal deve informar o usuário a respeito do erro de preenchimento. • Caso o valor das passagens para crianças seja diferente do valor das passagens para adultos, deve ser apresentada qual a regra utilizada, por exemplo: "Crianças acima de 6 (seis) anos devem pagar valor integral". RF02. RF03. (etc...)

## 7.2. REQUISITOS NÃO FUNCIONAIS

* O sistema deve ter uma interface web;
* Os componentes devem ser visíveis e bem identificados;
* Deve haver vídeos que orientam na operação do sistema;
* Uma interface de usuário simples de ser usada.

A efetivação da reserva do pacote, só deve ser liberada após o cliente estar logado no sistema. Informações: usuário e senha. Regras: o cliente terá acesso para comprar, consultar e alterar.

## 7.3. EVOLUÇÃO DO SISTEMA

O sistema baseado em plataforma WEB estará preparado para integrar as seguintes funcionalidades:

* Disponibilizar acesso através de plataforma móvel (smartphone);

## 7.4. DIAGRAMAS UML

## 7.4.1. DIAGRAMA DE SÊQUENCIAS

O diagrama de sequência mostra o usuário como ator, apresentando a sequência do castro de escola, dessa forma irá ter os componentes de negócios, validando se o cliente tem permissão, assim a checagem no banco de dados, retornando à validação

para criação de uma escola. Na segunda sequência os dados são gravados em sequência inseridos no banco, sendo redirecionado ao dash.

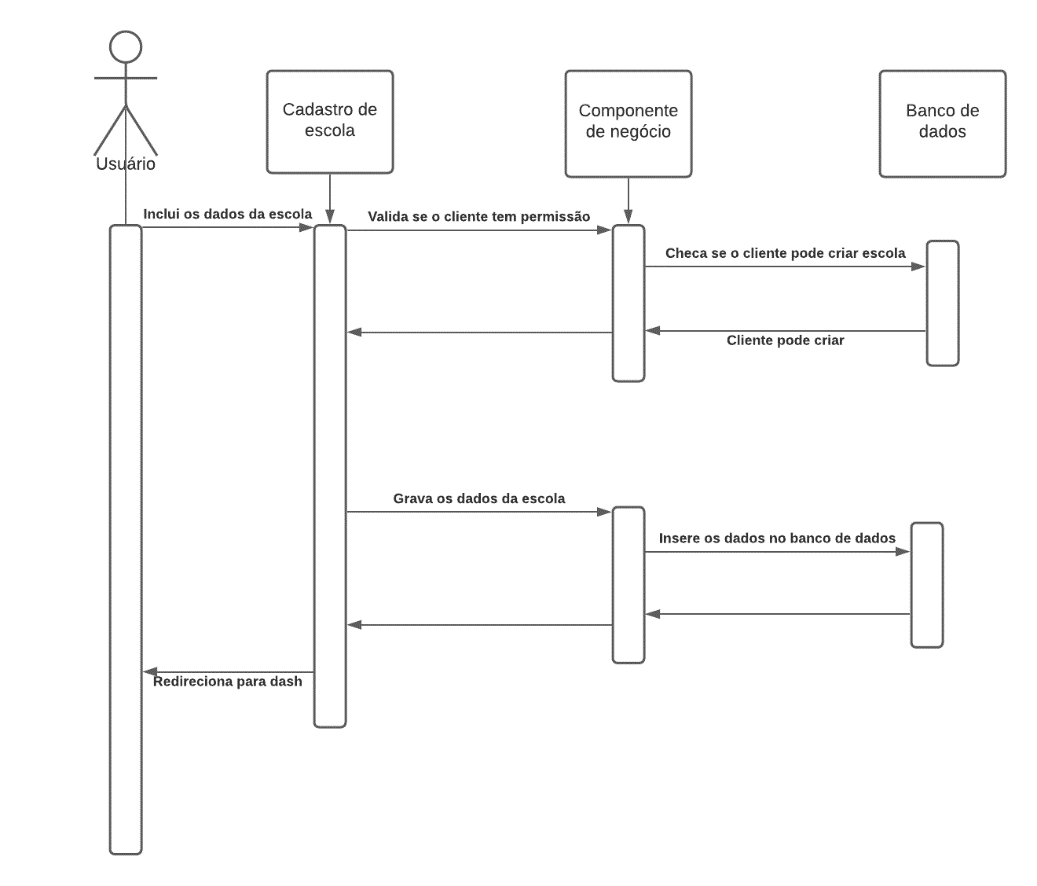


Figura 1 - Diagrama de sequências

## 7.4.2. DIAGRAMA DE CLASSES

O diagrama de classes mostra todas as classes usadas no sistema.

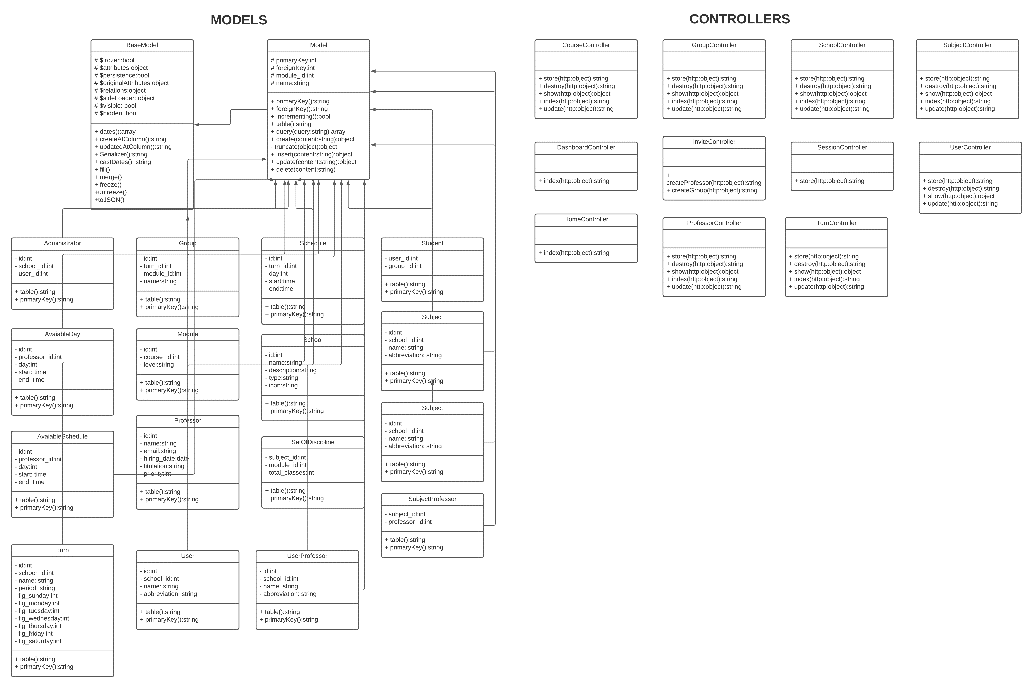


Figura 2 - Diagrama de classe

## 7.4.3. DIAGRAMA DE CASOS DE USO

O digrama de caso de uso apresenta as ações dos usuários(ator) no sistema.

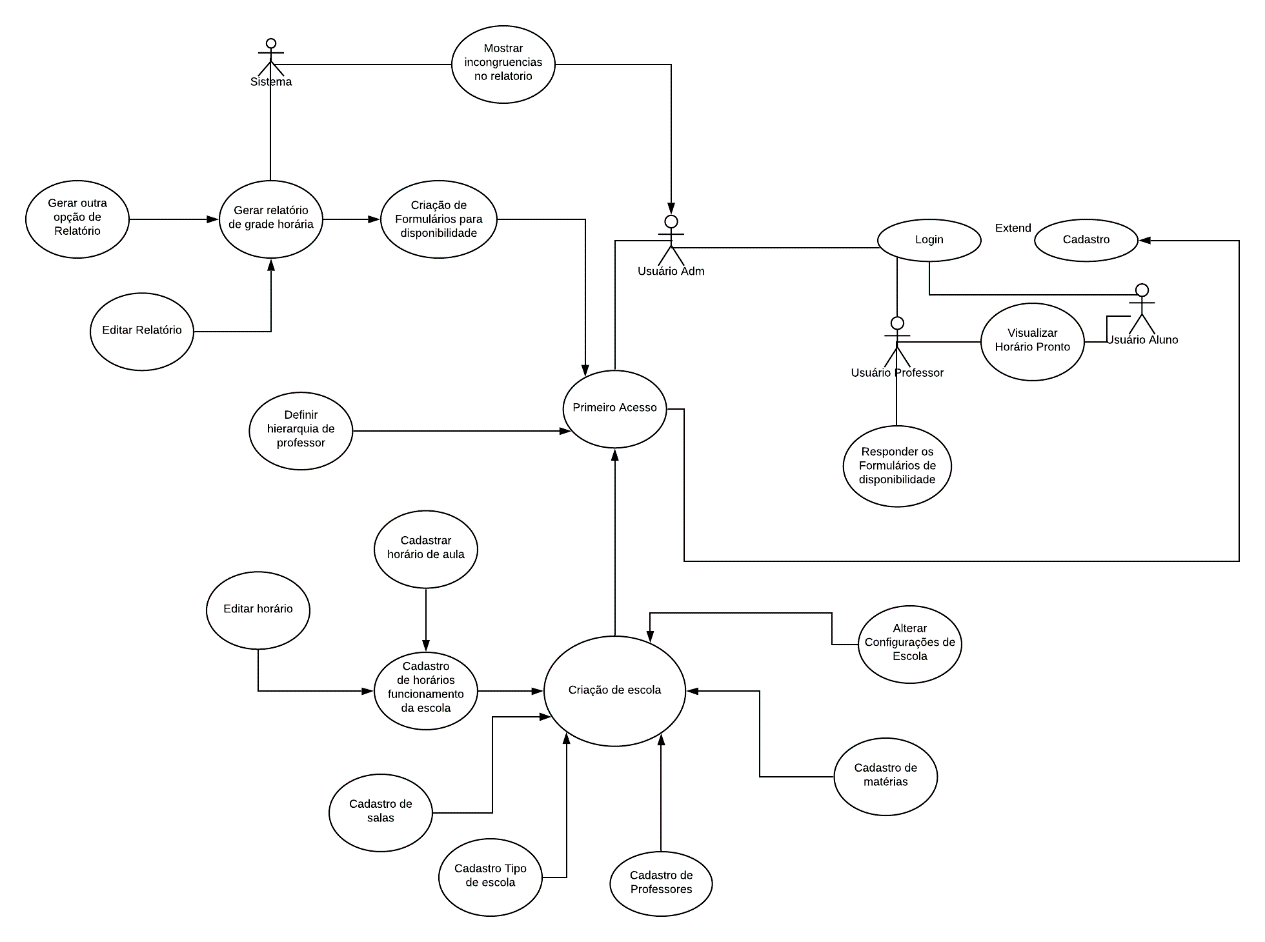


Figura 3 - Diagrama de casos de uso

## 7.4.4. DIAGRAMAS DE ATIVIDADES

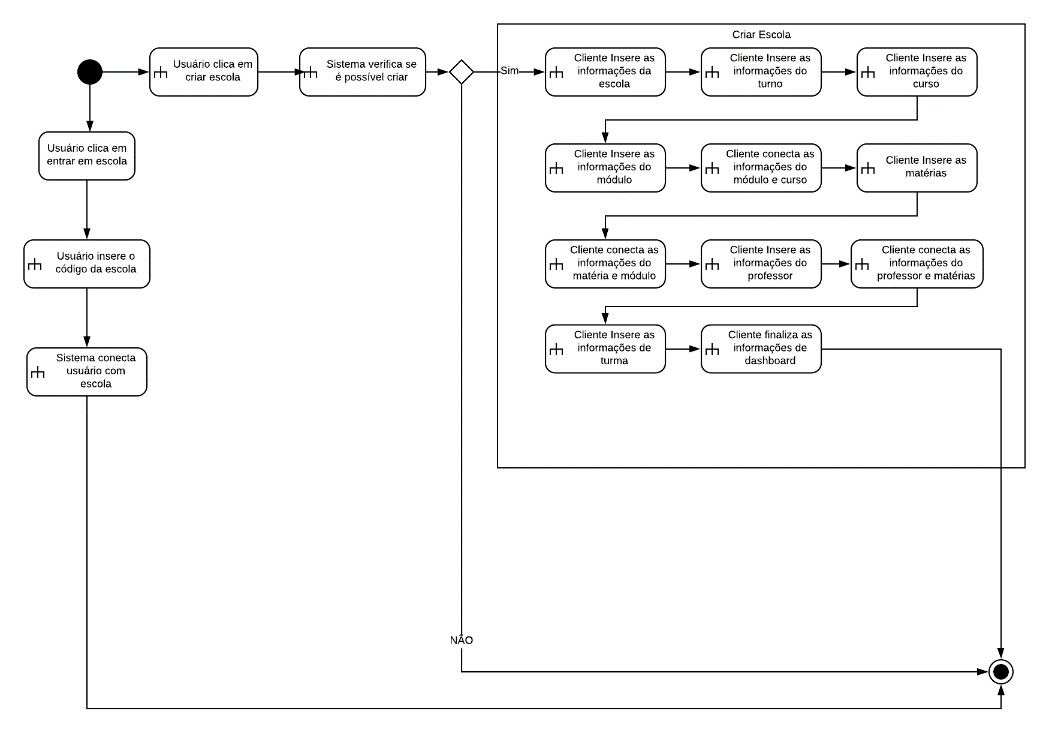
Nesse diagrama mostras todas as atividades que o sistema irá realizar em suas sequências.  


Figura 4 - Diagrama de atividades

## 7.4.5. DIAGRAMA ENTIDADE-RELACIONAMENTO (DER)

O DER apresenta toda a lógica e funcionamento do banco de dados do sistema.  
Diagrama

Descrição gerada automaticamente

Figura 5 - Diagrama Entidade-Relacionamento (DER)

# 8. FUNCIONALIDADES

Com análise dos dados obtidos foram desenvolvidas funcionalidades primordiais para o funcionamento integro do sistema.

## 8.1. AUTENTICAÇÃO

Cadastro;

A autenticação do projeto, conta com uma tela de cadastro para novos usuários e outras para as diferentes formas de login para usuários já cadastrados em nosso sistema, como login com seus dados pessoal, com o Google, Facebook ou Linkedin.

## 8.2. PRIMEIRO ACESSO

Concordo com os termos;

Escolher entre escuro ou claro;

Apresentação do Aplicativo;

Usuário/Cliente:

Escolher entre participar de um projeto ou cadastrar um projeto (Escola);

Administrador para designar cargos;

Cadastrar informações da empresa;

Ao se cadastrar pela primeira vez em nosso sistema, todos os novos usuários terão de passar por um primeiro acesso, onde serão direcionadas as telas de termos de uso, lendo e concordando serão direcionados a tela de escolha entre tema escuro ou claro da aplicação e por fim uma breve apresentação do sistema.

Após isso, os diferentes tipos de usuários/clientes poderão escolher entre participar de um projeto ou cadastrar um projeto (escola), administrar para designar cargos e cadastrar informações da empresa.

## 8.3. DASHBOARD

Últimos Relatórios;

Links para o cadastro.

A tela de Dashboard comporta informações pessoais de cada projeto (escola), bem como os últimos relatórios gerados e os links para cadastros.

## 8.4. CADASTRO

Importar informações de planilhas;

Cadastro de Professores;

Receber cadastro de um perfil ou cadastrar:

Nome;

E-mail;

Disponibilidade de horário;

Matéria(as) que ministra.

Cadastro de Matérias:

Nome;

Necessidade de laboratório;

Cadastro de Salas (espaços);

Cadastro de Turma (Ex: 3°A);

Cadastro de Horários:

Intervalos;

Tempo ocioso para limpeza e afins;

Na parte de cadastro será possível importar informações de planilhas, cadastrar de novos professores, receber cadastro de um perfil ou cadastrar, levando em considerações os seguintes dados: Nome; E-mail; Disponibilidade de horário; Matéria(as) que ministra. Cadastrar as matérias: Nome e Necessidade de laboratório. Cadastrar Salas (espaço), cadastrar turmas (Ex: 3°A) e por fim cadastrar horários, bem como seus intervalos e tempo ocioso para limpeza e afins.

## 8.5. RELATÓRIO

Os relatórios poderão ser emitidos, contendo em seu conteúdo a grade horária e divisão por cores.

## 8.6. PÓS RELATÓRIO

Será possível gerar um novo relatório, editá-lo ou ainda exportar nos formatos PDF e XLSX.

## 8.7. CONEXÕES

Conectar com o Google Agenda;

Conectar com o Microsoft Teams.

O cronograma de aula terá a flexibilidade de sua plataforma, podendo se conectar com a ferramenta do Microsoft Teams e ainda o Google Agendas.

## 8.8. CONFIGURAÇÃO

Tema Escuro;

Fale conosco;

Dúvidas Frequentes;

Divulgação por redes sociais;

Sugestões.

Nas configurações do programa, os usuários poderão alternar entre o tema do claro e escuro, entrar em contato com os desenvolvedores ou responsáveis pelo sistema, acessar as dúvidas frequentes e deixar sugestões em relação ao funcionamento.

## 8.9. ADICIONAIS

Alguns possíveis adicionais futuros para o projeto são: Chat Bot, para assistência e ajuda, um sistema de anúncio, bloqueio de AdBlock, enviar email para professores com o horário, adicionais dos adicionais (MVP), organizar matérias das aulas, tradução para inglês, aplicativo de celular, sugestões e pedidos.

# 9. ESPECIFICAÇÕES DO PROJETO

## 9.1 ESCOPO DO PROJETO

Todos os dados e lógicas de acordo ao funcionamento da Faculdade Lusófona de São Paulo, onde saiu a demanda de realização do sistema de criação de grade horária.  
 **LOGIN**

É possível realizar um cadastro e login com sua conta Google ou LinkedIn ou possivelmente e-mail institucional da faculdade;

**GERENCIAMENTO DE UMA OU MAIS ESCOLAS**

É possível criar uma identificação da escola por meio do nome e descrição

Se possível funcionalidade para os Funcionários Secretária.

**TURNO**

É possível criar/editar diferentes turnos com determinados horários especificando a duração da aula e duração do dia letivo.

Turnos: Noturno-matutino

**CURSOS**

Pode-se criar diferentes cursos para a mesma escola (exemplo: Regular, técnico etc.) com diversas séries dentro dele.

Gestão e Tecnologia:

* Administração;
* Ciências contábeis;
* Logística;
* Gestão financeira;
* Gestão em marketing;
* Gestão em RH.

Engenharias:

* Civil;
* Produção.

Saúde:

* Biomedicina;
* Enfermagem;
* Fisioterapia.

Licenciaturas:

* Pedagogia;
* Educação Física.

Outros:

* Direito;
* Gastronomia;
* Design de interiores;
* Psicologia.

**MATÉRIAS**

Para facilitar no cadastro, é possível cadastrar várias matérias e a quantidade de aulas de cada uma para várias séries de uma vez.

Disciplinas Comuns:

* Semestre único, primeiro, semestre para todos;
* Partir do segundo muda para as determinadas áreas.

**PROFESSORES**

O administrador pode colocar várias matérias para um mesmo professor e se precisar as matérias podem ter mais de um professor, podendo gerenciar cada série.

Na hora de gerar a grade horária, professores com maior prioridade pegarão horários "melhores".

* Atualmente são 62 docentes na faculdade;
* Coordenador define: Hierarquização e disponibilidade do horário;
* Opção prévia de visualização para o professor;
* Professor preenche com todas as informações e ou coordenadores faz a validação.

**TURMAS**

É possível também gerenciar diversas turmas com a carga horária que desejar.

* Quantidade de alunos;
* Cadastro de Salas, opções de salas.

**DASHBOARD**

Uma das funções do administrador é ter acesso ao painel de controle, nele você tem visibilidade de diversas informações a respeito da escola, turnos e salas.

**HOME**

Na tela principal é possível ver todas as escolas que o usuário possui e as respectivas turmas dentro dela.

**CONVITES**

Para que um aluno ou professor seja cadastrado na escola, é necessário convidá-lo a partir de convites específicos para suas respectivas funções.

**RELATÓRIOS**

Os relatórios são os registros das grades horários geradas de cada turma, podendo também ter a versão do professor, para os professores poderem facilmente ver onde darão suas aulas, os relatórios dos professores possibilita para o administrador visualizar os professores que estão em aula vaga, para quando houver um problema e algum professor faltar, o administrador já poder saber quem pode substituí-lo.

## 9.2. LIMITAÇÕES DO PROJETO

* Nosso sistema não é capaz de identificar salas físicas e diferir salas especiais.
* Somente é utilizado para o cadastro de escolas e geração de horários, não somos responsáveis por armazenar tarefas, faltas e outras coisas que não envolvem a grade horária.
* Não possui registro de professores substitutos.
* Não tem um recurso onde o professor pode avisar antecipadamente que irá faltar.
* Não guarda o número total de alunos de cada sala, o que guarda são os alunos cadastrados no sistema e conectados com as salas.
* Não tem dividimos as turmas.
* É uma aplicação de gerenciamento de grade horária e não de calendário
* Trabalhamos com o padrão de grade horária no Brasil, ou seja, durante um período letivo, determinadas matérias são postas e organizamos as matérias no determinado período.

# 11. TELAS PRINCIPAIS

TELA DE CADASTRO

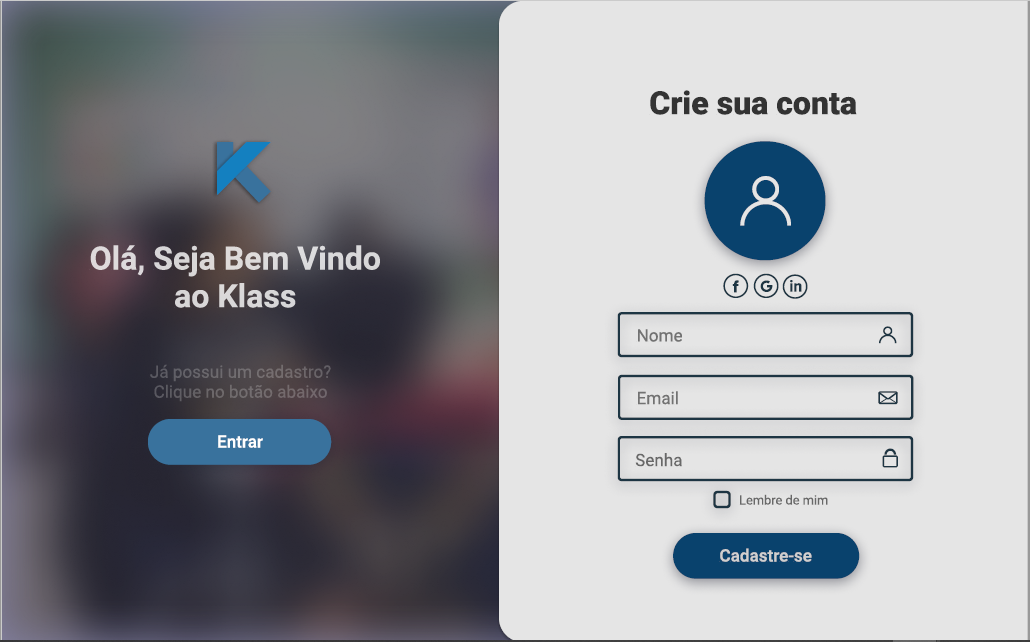


Figura 6 - Tela de Cadastro

Parte do código da tela de cadastro:

import React, { useContext, useEffect } from "react";  
  
import "./styles.css";  
import SocialMedias from "../../components/SocialMedias";  
import Input from "../../components/Input";  
import Checkbox from "../../components/Checkbox";  
import { useAlert } from "react-alert";  
import { Context } from "../../context/AuthContext";  
import api from "../../services/api";  
  
function Register() {  
 const { handleLogin } = useContext(Context);  
 const { register, handleSubmit, errors } = useForm();  
 const alert = useAlert();  
 const history = useHistory();  
  
 const onSubmit = async (data:any) => {  
 api.post('users', data).then((response)=>{  
 const { data: { token, name } } = response;  
 localStorage.setItem('token', JSON.stringify(token));  
 localStorage.setItem('username', JSON.stringify(name));  
 api.defaults.headers.Authorization = `Bearer ${token}`;  
 handleLogin()  
 history.push('/access');  
 }).catch(error => {  
if(!error.response) alert.error("Impossível conectar ao servidor!");  
 else alert.error(error.response.data[0].message);

});  
 };

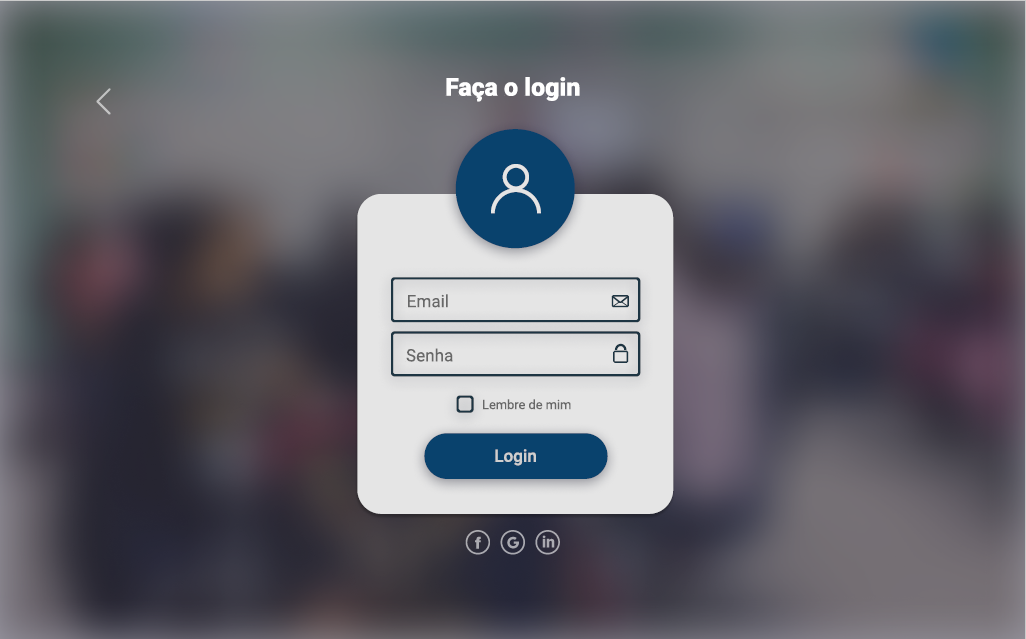
TELA DE LOGIN  


Figura 7 - Tela de Login

Parte do código da tela de login:

import React, { useContext, useEffect } from "react";  
import { useHistory } from "react-router-dom";  
  
import "./styles.css";  
import { KaUser, KaMail, KaPassword } from "../../assets/icons";  
import Input from "../../components/Input";  
  
function Login() {  
 const { handleLogin } = useContext(Context);  
 const { register, handleSubmit, errors } = useForm();  
 const alert = useAlert();  
 const history = useHistory();  
   
 const onSubmit = async (data:any) => {  
 api.post('sessions', data).then((response)=>{  
 const { data: { token, name } } = response;  
 localStorage.setItem('token', JSON.stringify(token));  
 localStorage.setItem('username', JSON.stringify(name));  
 api.defaults.headers.Authorization = `Bearer ${token}`;  
 handleLogin()  
 history.push('/home');  
 }).catch(error => {  
 if(!error.response) alert.error("Impossível conectar ao servidor!");  
 else if (!error.response.data[0]) alert.error("Preencha os campos!");  
 else alert.error(error.response.data[0].message);  
 });  
 };  
   
export default Login;

TELA DE ESCOLHA DE TEMA



Figura 8 – Tela Escolha de Tema

Essa tela mostra duas opções de uso temáticos do sistema, escuro e claro.

Parte do código da tela de temas:

import React, {createContext, useState, useContext, Dispatch, SetStateAction,useEffect, } from "react";  
import { lightTheme, darkTheme } from "../assets/styles/css-vars";  
  
type Dispatcher<S> = Dispatch<SetStateAction<S>>;  
  
const setThemeProps: Dispatcher<string> = () => null;  
  
const ThemeContext = createContext({  
 theme: "",  
 setTheme: setThemeProps,  
});  
  
const ThemeProvider: React.FC = ({ children }) => {  
 const [theme, setTheme] = useState("light");  
  
 useEffect(() => {  
 const currentTheme = localStorage.getItem("theme");  
 if (currentTheme) setTheme(currentTheme);  
 }, []);  
  
 useEffect(() => {  
 const cssVars = theme === "dark" ? darkTheme : lightTheme;  
 cssVars.forEach((color) => {  
 document.documentElement.style.setProperty(color.name, color.value);  
 });  
 localStorage.setItem("theme", theme);  
 }, [theme]);  
  
 return (  
 <ThemeContext.Provider  
 value={{  
 theme,  
 setTheme,  
 }}  
 >  
 {children}  
 </ThemeContext.Provider>  
 );  
};

TELA E-MAIL DE ATIVAÇÃO

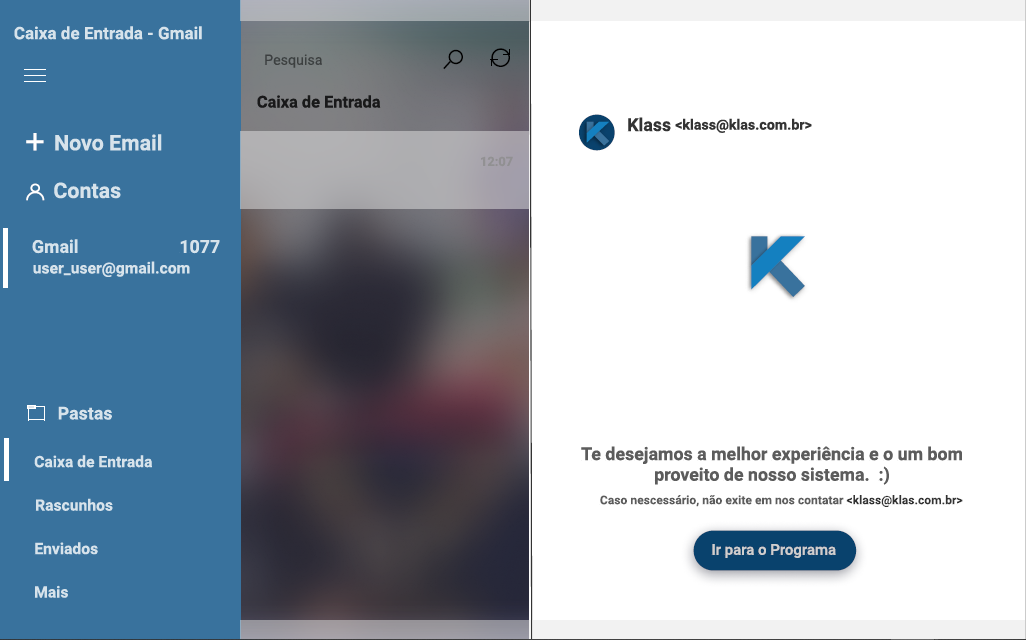


Figura 9 - Tela e-mail de ativação

TELA HOME



Figura 10 - Tela Home

Parte do código da tela home:

import { getColor } from "../../utils/colors";  
import InitialCreateSchool from "../../components/Modals/InitialCreate";  
  
function Home() {  
 const [activeInitial, setActiveInitial] = useState(false);  
 const [activeMenu, setActiveMenu] = useState(false);  
 const [activeEnterSchool, setActiveEnterSchool] = useState(false);  
 const [schools, setSchools] = useState([]);  
  
 useEffect(() => {  
 (async () => {  
 const { data } = await api.get('/home');  
 setSchools(data);  
 })()  
 }, []);  
}

TELA DE CRONOGRAMA

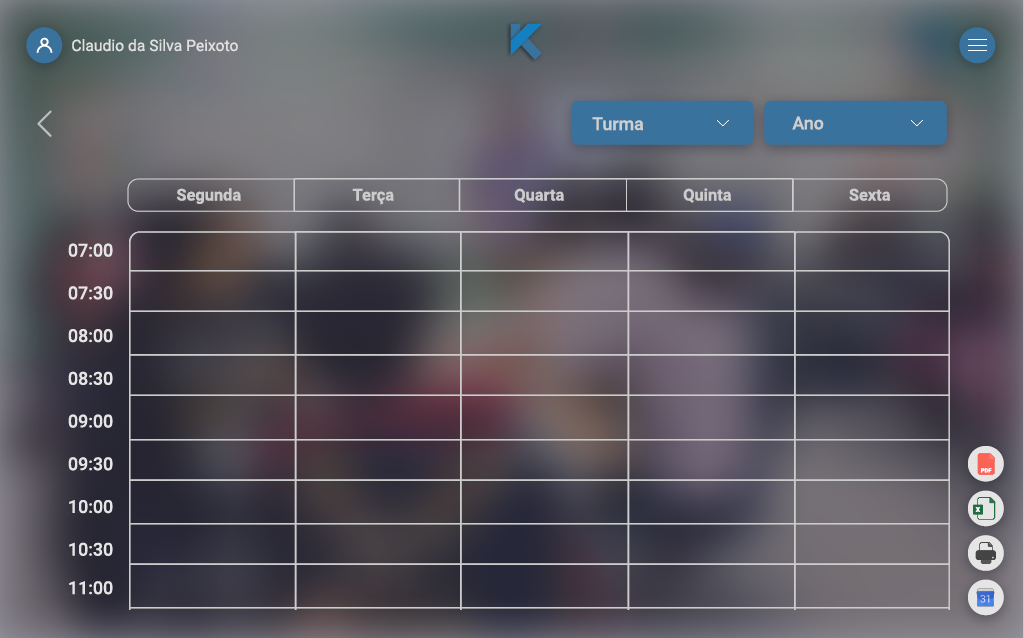


Figura 11 - Tela de Cronograma

Parte do código da tela de Cronograma:

interface ScheduleProps {  
 sideContainerSize ?: number;  
 to?: string;  
}  
  
const Schedule:React.FC<ScheduleProps> = ({sideContainerSize = 0, to = "/home"}) => {  
 const [fitRows, setFitRows] = useState(7);  
 const [maxRowQuantity, setMaxRowQuatity] = useState(7);  
 const [minRowQuantity, setMinRowQuatity] = useState(0);  
 const [sendTo, setSendTo] = useState(to);

React.useEffect(() => {  
 function handleResize() {  
 let maxQuantity;  
 if (window.innerWidth > (1205+sideContainerSize)) maxQuantity = 7;  
 else if(window.innerWidth > (1045+sideContainerSize)) maxQuantity = 6;  
 else if(window.innerWidth > (885+sideContainerSize)) maxQuantity = 5;  
 else if(window.innerWidth > (725+sideContainerSize)) maxQuantity = 4;  
 else if(window.innerWidth > (560+sideContainerSize)) maxQuantity = 3;  
 else if(window.innerWidth > (405+sideContainerSize)) maxQuantity = 2;  
 else maxQuantity = 1;  
 if(maxQuantity < (maxRowQuantity - minRowQuantity)) setMaxRowQuatity(maxQuantity - minRowQuantity);  
 setFitRows(maxQuantity);  
 }  
 handleResize();  
 window.addEventListener('resize', handleResize);  
 if(to !== '/home') {  
 const search = window.location.search;  
 const params = new URLSearchParams(search);  
 const id = params.get('id');  
 setSendTo(sendTo+'?id='+id)  
 }  
 }, [])

export default Schedule;

TELAS DO PROCESSO DE CRIAÇÃO DE ESCOLA



Figura 12 - Tela criação nova escola

Parte do código da tela inicial de criação de escola:

import React, { useState } from "react";  
import { SwitchTransition, CSSTransition } from "react-transition-group";  
  
import "./styles.css";  
import Create1 from "./components/Create1";  
import ExitCreateSchool from "./components/Exit";  
  
function CreateSchool() {  
 const [activeExit, setActiveExit] = useState(false);  
 const [activeSuccessful, setActiveSuccessful] = useState(false);  
 const [step, setStep] = useState(0);  
 const [mode, setMode] = useState('foward');  
 const alert = useAlert();  
 const history = useHistory();  
  
 const [schoolName, setSchoolName] = useState('');  
 const [schoolDescription, setSchoolDescription] = useState('');  
 const [schoolType, setSchoolType] = useState('');  
 const [selectedImg, setSelectedImg] = useState('');  
 const arrtype:Array<any> = [];  
 const [turns, setTurns] = useState(arrtype);  
 const [courses, setCourses] = useState(arrtype);  
 const [modules, setModules] = useState(arrtype);  
 const [subjects, setSubjects] = useState(arrtype);  
 const [teachers, setTeachers] = useState(arrtype);  
 const [classes, setClasses] = useState(arrtype);  
  
 async function createTurns(school\_id: string) {  
 await Promise.all(turns.map(async (turn:any, ind) => {  
 const start = turn.content.schedule.split(' às ')[0];  
 const end = turn.content.schedule.split(' às ')[1];  
 const newIntervals:any = [];  
 turn.content.intervals.map(({title}:any)=> {  
 const startInterval = title.split(' às ')[0];  
 const endInterval = title.split(' às ')[1];  
 newIntervals.push({start:startInterval, end: endInterval});  
 })  
 const turnRes = await api.post(`/schools/${school\_id}/turns`, {  
 name: turn.title,  
 start,  
 end,   
 class\_duration: turn.content.classDuration,  
 week\_days: turn.content.days,  
 intervals: newIntervals,  
 });  
  
 let newTurns:any = turns;  
 const newTurn = turn;  
 newTurn.id = turnRes.data.id;  
 newTurns[ind] = newTurn;  
 setTurns([...newTurns]);  
 }));  
 }

TELA INSERÇÃO DE INFORMAÇÕES DA ESCOLA

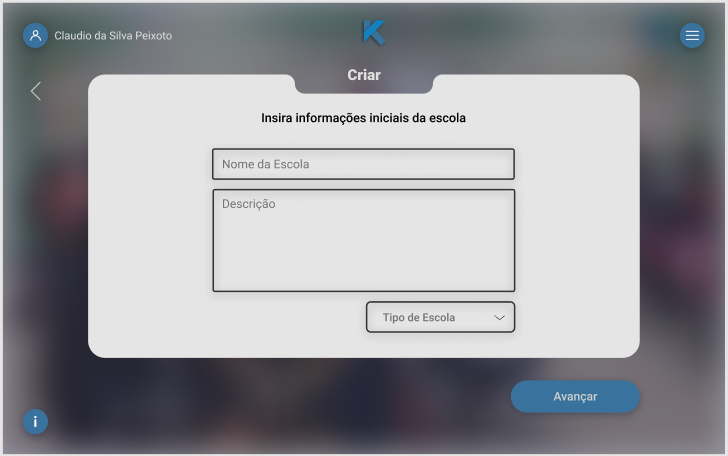


Figura 13 - Tela de nome, descrição e de tipo de escola

Parte do código da Tela de nome, descrição e de tipo de escola:

return (  
 <TruncatedContainer title="Criar" className="create-school-container">  
 <div className="create-school-content">  
 <h1>Insira informações iniciais da escola</h1>  
 <div className="input-container">  
 <input type="text" placeholder="Nome da escola" value={name} onChange={e => setName(e.target.value)} />  
 <textarea placeholder="Descrição da escola" value={description} onChange={e => setDescription(e.target.value)} />  
 <select value={type} onChange={e => setType(e.target.value)}>  
 <option value="" disabled>  
 Tipo de escola  
 </option>  
 <option value="1">Faculdade</option>  
 </select>  
 </div>  
 </div>  
 </TruncatedContainer>  
);

TELA INSERÇÃO DE ÍCONE



Figura 14 - Tela escolha de ícone para a escola

Parte do código da Tela escolha de ícone da escola:

function handleSelectPic(img:string) {  
 setSelectedImg(img);  
}  
  
return (  
 <TruncatedContainer title="Criar" className="create-school-container">  
 <div className="create-school-content">  
 <h1>Escolha um icone</h1>  
 <div className="icons-container">  
 <div className="scroll-view">  
 <div>  
 <div className="icons-content">  
 {cards.map((Card, ind) => (  
 <img  
 key={ind}  
 style={{cursor: 'pointer'}}  
 src={Card}  
 height={100}  
 onClick={() => handleSelectPic(Card)}  
 />  
 ))}  
 </div>  
 <span>  
 Ícones feitos por <a href="https://www.flaticon.com/br/autores/dighital" title="Dighital" target="\_blank">Dighital</a> de <a href="https://www.flaticon.com/br/" title="Flaticon" target="\_blank"> www.flaticon.com</a>  
 </span>  
 </div>  
 </div>  
 <div className="icon-description">  
 <ContentCard  
 text={schoolName}  
 color="#F68237"  
 title=""  
 img={selectedImg}  
 />  
 </div>  
 </div>  
 </div>  
 </TruncatedContainer>  
);

TELA CRIAÇÃO DE TURNOS

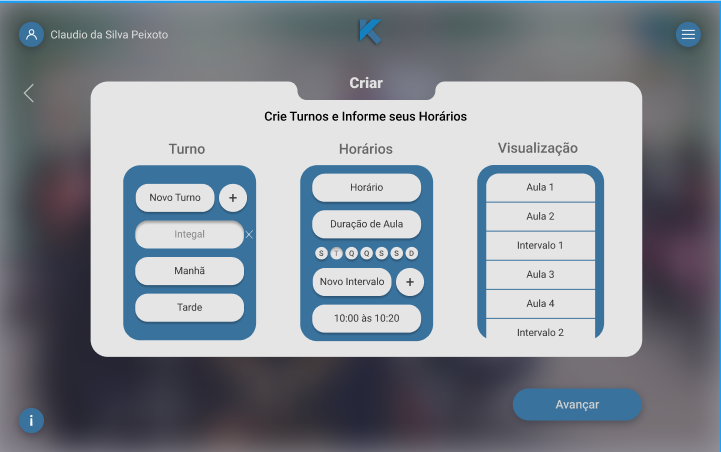


Figura 15 - Tela criação turnos e horários

Parte do código da tela de criação de turnos:

return (  
 <TruncatedContainer title="Criar" className="create-school-container">  
 <div className="create-school-content">  
 <h1>Crie Turnos e Informe seus Horários</h1>  
 <div className="creation-group">  
 <div className="creation-container">  
 <div className="creation-header">  
 <h1>Turno</h1>  
 </div>  
 <CreateCardSection  
 cards={turns}  
 addCard={addTurn}  
 removeCard={removeTurn}  
 selectedCard={selectedTurn}  
 setSelectedCard={setSelectedTurn}  
 placeholder="Novo Turno"  
 />  
 </div>  
 <div className="creation-container">  
 <div className="creation-header">  
 <h1>Horários</h1>  
 </div>  
 <TurnInfoContainer  
 addInterval={addInterval}  
 removeInterval={removeInterval}  
 selectedTurn={selectedTurn}  
 turns={turns}  
 setTurns={setTurns}  
 />  
 </div>  
 <div className="creation-container">  
 <div className="creation-header">  
 <h1>Visualização</h1>  
 </div>  
 <div className="creation-content">  
 <List   
 duration={turns[selectedTurn]?.content?.classDuration || 0}   
 schedule={turns[selectedTurn]?.content?.schedule || ''}   
 intervals={turns[selectedTurn]?.content?.intervals || []}   
 />  
 </div>  
 </div>  
 </div>  
 </div>  
 </TruncatedContainer>  
 );  
};

TELA CRIAÇÃO DE CURSOS E MÓDULOS

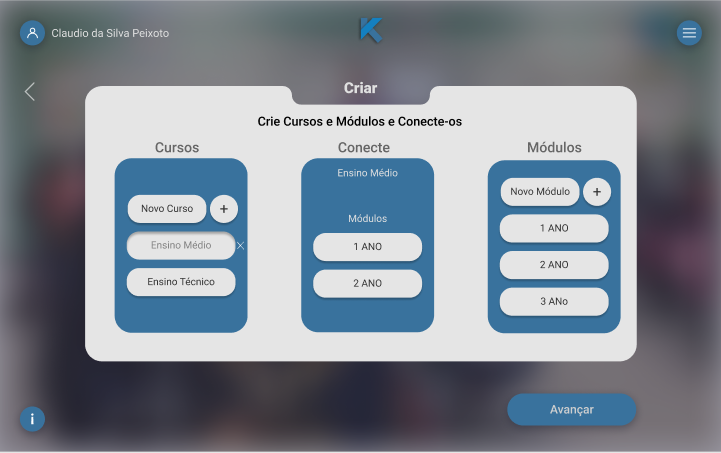


Figura 16 - Tela criação de cursos e módulos

Parte código tela de criação de cursos e módulos:

<div className="create-school-content">  
 <h1>Crie Cursos, Módulos, Conecte-os</h1>  
 <div className="creation-group">  
 <div className="creation-container">  
 <div className="creation-header">  
 <h1>Cursos</h1>  
 </div>  
 <CreateCardSection  
 cards={courses}  
 addCard={addCourse}  
 removeCard={removeCourse}  
 selectedCard={selectedCourse}  
 setSelectedCard={setSelectedCourse}  
 placeholder="Novo Curso"  
 />  
 </div>  
 <DragDropContext onDragEnd={onDragEnd}>  
 <ConnectionSection  
 cards={courses}  
 selectedCard={selectedCourse}  
 removeConnection={removeConnection}  
 unselectedMessage="Selecione um curso"  
 noCardMessage="Insira um módulo"  
 title="Módulos"  
 />  
 <div className="creation-container">  
 <div className="creation-header">  
 <h1>Módulos</h1>  
 </div>  
 <CreateDraggableSection  
 cards={modules}  
 addCard={addModule}  
 removeCard={removeModule}  
 placeholder="Novo Módulo"  
 />  
 </div>  
 </DragDropContext>  
 </div>  
 </div>

TELA CRIAÇÃO DE MATÉRIAS

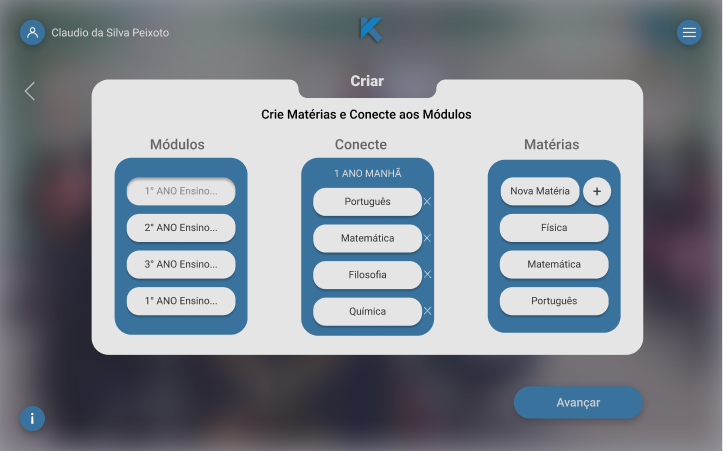


Figura 17 - Tela criação de Matérias e Conexão aos módulos

Parte do código de criação matérias:

<div className="create-school-content">  
 <h1>Crie Matérias, Conecte com seus Módulos</h1>  
 <div className="creation-group">  
 <div className="creation-container">  
 <div className="creation-header">  
 <h1>Módulos</h1>  
 </div>  
 <CardSection  
 cards={modules}  
 selectedCard={selectedModule}  
 setSelectedCard={setSelectedModule}  
 />  
 </div>  
 <DragDropContext onDragEnd={onDragEnd}>  
 <ConnectionSection  
 cards={modules}  
 setCards={setModules}  
 selectedCard={selectedModule}  
 removeConnection={removeConnection}  
 unselectedMessage="Selecione um módulo"  
 noCardMessage="Insira uma matéria"  
 title="Matérias"  
 withNumber  
 />  
 <div className="creation-container">  
 <div className="creation-header">  
 <h1>Matérias</h1>  
 </div>  
 <CreateDraggableSection  
 cards={subjects}  
 addCard={addSubject}  
 setCards={setSubjects}  
 removeCard={removeSubject}  
 placeholder="Nova Matéria"  
 withNumber  
 />  
 </div>  
 </DragDropContext>  
 </div>  
 </div>

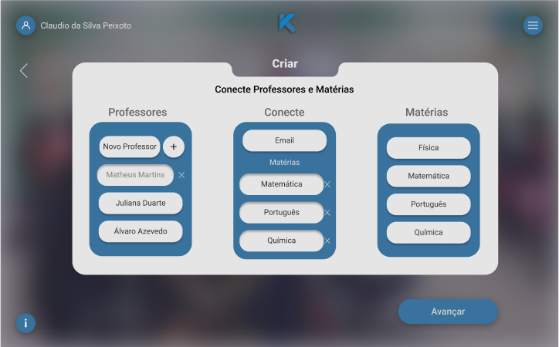


Figura 18 - Criação Professores e Matérias

Parte do código da tela de criação professores e matérias:

<div className="create-school-content">  
 <h1>Crie Matérias, Conecte com seus Módulos</h1>  
 <div className="creation-group">  
 <div className="creation-container">  
 <div className="creation-header">  
 <h1>Professores</h1>  
 </div>  
 <CreateCardSection  
 cards={teachers}  
 removeCard={removeTeacher}  
 addCard={addTeacher}  
 selectedCard={selectedTeacher}  
 setSelectedCard={setSelectedTeacher}  
 placeholder="Novo Professor"  
 />  
 </div>  
 <DragDropContext onDragEnd={onDragEnd}>  
 <ConnectionSection  
 cards={teachers}  
 selectedCard={selectedTeacher}  
 removeConnection={removeConnection}  
 unselectedMessage="Selecione um professor"  
 noCardMessage="Insira uma matéria"  
 title="Matérias"  
 email={selectedTeacher !== -1?teachers[selectedTeacher].email:undefined}  
 setEmail={changeEmail}  
 />  
 <div className="creation-container">  
 <div className="creation-header">  
 <h1>Matérias</h1>  
 </div>  
 <DraggableSection cards={subjects} />  
 </div>  
 </DragDropContext>  
 </div>  
 </div>

TELA DE ATRIBUIÇÃO DE TURNOS AOS MÓDULOS

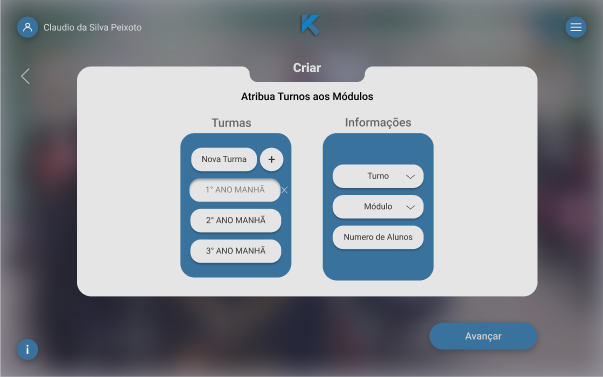


Figura 19 - Tela de atribuição turnos aos módulos

Parte do código da tela de turnos:

<div className="create-school-content">  
 <h1>Crie Turmas</h1>  
 <div className="creation-group">  
 <div className="creation-container">  
 <div className="creation-header">  
 <h1>Turmas</h1>  
 </div>  
 <CreateCardSection  
 cards={classes}  
 removeCard={removeClass}  
 addCard={addClass}  
 selectedCard={selectedClass}  
 setSelectedCard={setSelectedClass}  
 placeholder="Nova Turma"  
 />  
 </div>  
 <div className="creation-container">  
 <div className="creation-header">  
 <h1>Informações</h1>  
 </div>  
 <ClassInfoContainer  
 turns={turns}  
 modules={modules}  
 classes={classes}  
 setClasses={setClasses}  
 selectedClass={selectedClass}  
 />  
 </div>  
 </div>  
 </div>

Após isso aparece uma notificação de criado. Sendo essa a sequência de telas seguidas na criação de uma escola no sistema Klass.

TELA DE DASHBOARD



Figura 20 - Tela de dashboard

Parte do código da tela de Dashboard:

import React, { useEffect, useState } from "react";  
import "./styles.css";  
import api from "../../../services/api";  
import BackButton from "../../../components/BackButton";  
  
  
function Dashboard(){  
 const history = useHistory();  
 const dashboardBlank:any = {};  
 const [dashboardData, setDashboardData] = useState(dashboardBlank);  
  
 useEffect(() => {  
 (async () => {  
 const search = window.location.search;  
 const params = new URLSearchParams(search);  
 const id = params.get('id');  
 if(!id) history.push('/home');  
 try {  
 const dashboard = await api.get(`/schools/${id}/dashboard`);  
 setDashboardData({...dashboard.data.dashboard, id});  
 }  
 catch {  
 history.push('/home');  
 }  
 })()  
 }, []);

TELA DE PÁGINA DE ERRO



Figura 21 - Tela erro na página

Parte do código da tela de erro:Parte superior do formulário

import React from 'react';  
import BackButton from '../../components/BackButton';  
import './styles.css';  
  
  
const Page404: React.FC = () => {  
 return (  
 <div className="main-404">  
 <BackButton to="/" />  
 <div className="content-404">  
 <h1>404</h1>  
 <h2>Página não encontrada</h2>  
 </div>  
 </div>  
 );  
}

TELA ENTRAR EM UMA ESCOLA



Figura 22 - Tela participação de uma escola

Parte do código da Tela de entrar em uma escola:

const EnterSchool:React.FC<EnterSchoolProps> = ({active, setActive}) => {  
 const [code, setCode] = useState('');  
  
  
 async function handleEnter() {  
 const response = await api.post('/invites/students', {group\_id: code});  
 setCode('');  
 setActive(false)  
 }  
  
 return (  
 <ModalContainer active={active} setActive={setActive} title="Participar de uma nova escola">  
 <input   
 name="school-id"   
 placeholder="Digite o Código de Escola"   
 className="enter-school-input"   
 value={code}  
 onChange={e => setCode(e.target.value)}  
 />  
 <div className="enter-school-buttons">  
 <button className="btn-small" onClick={()=> setActive(false)}>Cancelar</button>  
 <button className="btn-small" onClick={()=> handleEnter()}>Participar</button>  
 </div>  
 </ModalContainer>  
 )

PRIMEIRO ACESSO



Figura 23 - Tela primeiro acesso

Segue parte do código da tela de primeiro acesso:

import React, { useState } from "react";  
  
import "./styles.css";  
import { KaCircleOutline, KaCircleSelected } from "../../assets/icons";  
import BackButton from "../../components/BackButton";  
  
function FirstAccess() {  
 const [step, setStep] = useState(0);  
 const [mode, setMode] = useState("foward");  
 const history = useHistory();  
 const { setTheme } = useTheme();  
  
 function newThemeScreen() {  
 return <ThemeScreen setTheme={setTheme} />;  
 }  
  
 const screens = [TermsScreen, newThemeScreen, PresentationScreen];  
  
 function handleNext() {  
 if (step === 2) history.push("/home");  
 setMode("foward");  
 setStep(step + 1);  
 }  
 function handleBack(e: any) {  
 e.preventDefault();  
 setMode("backward");  
 setStep(step - 1);  
 }  
  
export default FirstAccess;

VÍDEO TUTORIAL

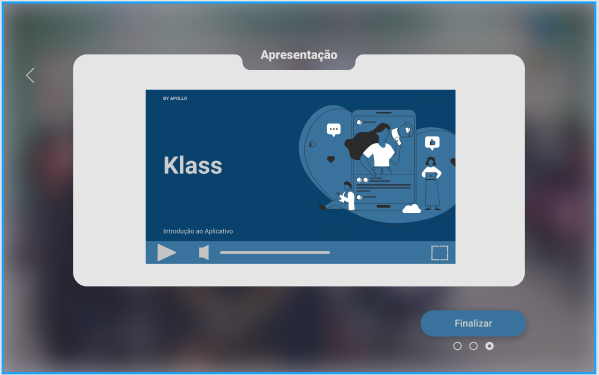


Figura 24 - Tela de vídeo tutorial

Parte do código da tela de tutorial:

import './styles.css';  
import TruncatedContainer from '../../components/TruncatedContainer';  
import BackButton from '../../components/BackButton';  
  
function TutorialCreate() {  
 return(  
 <div className="tutorial-create-container">  
 <BackButton to="/home" />  
 <TruncatedContainer title="Tutorial Criar">  
 <YouTube className="video7" videoId="Zc1OOS4aMbU" />  
 </TruncatedContainer>  
 </div>  
 )  
}

INFORMAÇÕES DO PERFIL

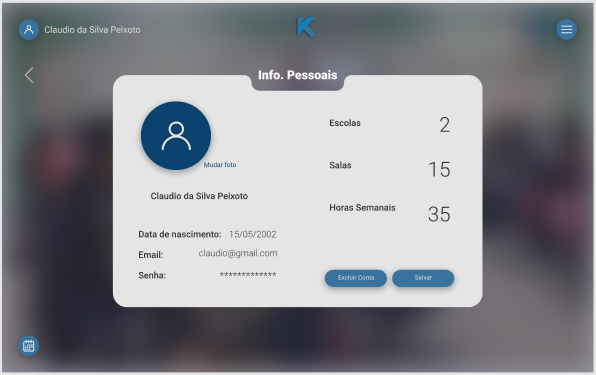


Figura 25 - Tela de perfil

Parte do código da tela de perfil:

import "./styles.css";  
import TruncatedContainer from "../../components/TruncatedContainer";  
import BackButton from "../../components/BackButton";  
  
function Profile() {  
 const [active, setActive] = useState(false);  
 const [icon, setIcon] = useState('');  
 const [username, setUsername] = useState('');  
 const [birthDate, setBirthDate] = useState('');  
 const [email, setEmail] = useState('');  
 const [password, setPassword] = useState('\*\*\*\*\*\*\*\*\*\*\*');  
  
 useEffect(() => {  
 (async()=> {  
 const {data}: any = await api.get('/users');  
 setUsername(data.user.name);  
 setBirthDate(data.user.birth\_date);  
 setEmail(data.user.email);  
 setIcon(data.user.icon);  
 localStorage.setItem('icon', data.user.icon);  
 })()  
 }, [active])

TELA DE TURMAS



Figura 26 - Tela de turmas

Parte do código da tela de turmas:

return (  
 <div className="classes-container">  
 <BackButton to={`/dashboard?id=${schoolId}`} />  
 <div className="scroll-view">  
 {turns.map(({ name }:any, ind:number) => (  
 <div key={ind} className="classes-cards-container">  
 <h1>{name}</h1>  
 <div className="classes">  
 {groups.map(({ name, group\_id }: any, ind:any) => (  
 <ContentCard  
 key={ind}  
 title={name.substr(0, 1) + name.substr(-1, 1)}  
 text={name}  
 color={getColor(name, ind)}  
 onClick={() => handleClick(group\_id)}  
 style={{ cursor: "pointer" }}  
 small  
 />  
 ))}  
 <ContentCard  
 title="+"  
 text="Adicionar"  
 color={colors[18]}  
 onClick={() => {}}  
 style={{ cursor: "pointer" }}  
 small  
 />  
 </div>  
 </div>  
 ))}  
 </div>  
 <InfoModal  
 active={active}  
 setActive={setActive}  
 handleDelete={() => {}}  
 handleSave={() => {}}  
 fields={[  
 {  
 name: "Nome:",  
 value: gradeName,  
 setValue: setGradeName  
 },  
 {  
 name: "Alunos:",  
 value: gradeStudents,  
 setValue: setGradeStudents  
 },  
 {  
 name: "Turno:",  
 value: gradeTurn,  
 setValue: setGradeTurn  
 },  
 {  
 name: "Módulo:",  
 value: gradeCourse,  
 setValue: setGradeCourse  
 }  
 ]}  
 title={gradeName.substr(0,1)+gradeName.substr(-1,1)}  
 subtitle={gradeName}  
 />  
 </div>  
 );

Ao clicar em cada ícone aparecer uma pequena tela mostrando as informações de cada disciplina

TELAS DE CURSOS



Figura 27 - Telas de cursos

Parte do código da tela de cursos:

return (  
 <div className="main-courses">  
 <BackButton to={`/dashboard?id=${schoolId}`} />  
 <TruncatedContainer title="Cursos">  
 <div className="scroll-bar">  
 <div className="courses-container">  
 {courses.map(({name}:any) => (  
 <div className="courses-content">  
 <h1>{name}</h1>  
 <div className="select-container">  
 <select>  
 <option value="0">Integral</option>  
 <option value="1">Manhã</option>  
 <option value="2">Tarde</option>  
 <option value="3">Noite</option>  
 </select>  
 </div>  
 <div className="teacher-container">   
 <h1>Professores</h1>  
 <h1>10</h1>  
 </div>  
 <div className="icon-container">  
 <div className="circle-icon-container">  
 <KaUser size={15} color="var(--color-border-primary)"/>  
 </div>   
 </div>  
 <div className="matter-container">   
 <h1>Matérias</h1>  
 <h1>11</h1>  
 </div>  
 <div className="icon-container">  
 <div className="circle-icon-container">  
 <KaBook size={18} color="var(--color-border-primary)"/>  
 </div>   
 </div>  
 <h1>7:30 - 15:30</h1>  
 </div>  
 ))}  
  
 </div>  
 </div>  
 </TruncatedContainer>  
 </div>  
 )

TELA DE TURNOS

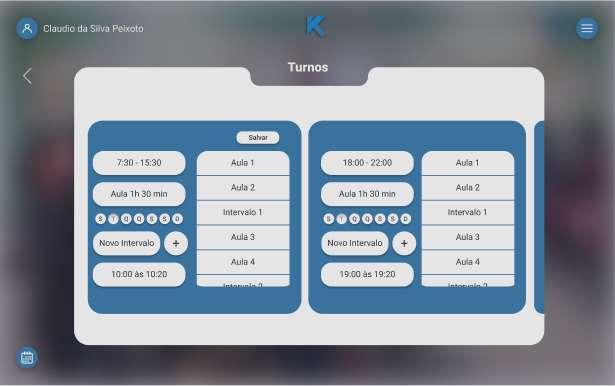


Figura 28 - Tela de turnos

Parte do código da tela de turnos:

import React, { useEffect, useState } from 'react';  
import { useHistory } from 'react-router-dom';  
import { isGetAccessor } from 'typescript';  
  
 const {  
 addInterval,  
 removeInterval,  
 } = create3Functions(turns, setTurns, 0);  
   
 return (  
 <div className="main-turns">  
 <BackButton to={`/dashboard?id=${schoolId}`} />  
 <TruncatedContainer title="Cursos">  
 <div className="scroll-bar">  
 <div className="turns-container">  
 {turns.map((turn:any, ind:number)=> (  
 <div className="turns-content">  
 <h1>Integral</h1>  
 <div className="creation-group">  
 <div className="creation-container">  
 <TurnInfoContainer  
 addInterval={addInterval}  
 removeInterval={removeInterval}  
 selectedTurn={ind}  
 turns={turns}  
 setTurns={setTurns}  
 />  
 </div>  
 <div className="creation-container">  
 <List   
 duration={turns[ind]?.content?.classDuration || 0}   
 schedule={turns[ind]?.content?.schedule || ''}   
 intervals={turns[ind]?.content?.intervals || []}   
 />  
 </div>  
 </div>  
 </div>  
 ))}  
  
 </div>  
 </div>  
 </TruncatedContainer>  
 </div>  
 );  
  
TELA DE MÓDULOS  
  


Figura 29 - Tela de módulos

Parte do código da tela de módulos:  
  
function Modules() {  
 return (  
 <div className="main-modules">  
 <BackButton to="/dashboard" />  
 <TruncatedContainer title="Módulos">  
 <div className="scroll-bar">  
 <div className="modules-container">  
 <div className="modules-content">  
 <h1>1° Ano</h1>  
 <div className="class-container">  
 <h1>Aulas</h1>  
 <h1>140</h1>  
 </div>  
 <div className="teacher-container">  
 <h1>Professores</h1>  
 <h1>10</h1>  
 </div>  
 <div className="icon-container">  
 <div className="circle-icon-container">  
 <KaUser size={15} color="var(--color-border-primary)"/>  
 </div>  
 <div className="circle-icon-container">  
 <KaUser size={15} color="var(--color-border-primary)"/>  
 </div>  
 <div className="circle-icon-container">  
 <KaUser size={15} color="var(--color-border-primary)"/>  
 </div>  
 <div className="circle-icon-container">  
 <KaUser size={15} color="var(--color-border-primary)"/>  
 </div>  
 <div className="circle-icon-container">  
 <KaUser size={15} color="var(--color-border-primary)"/>  
 </div>

TELA DE DOCENTES

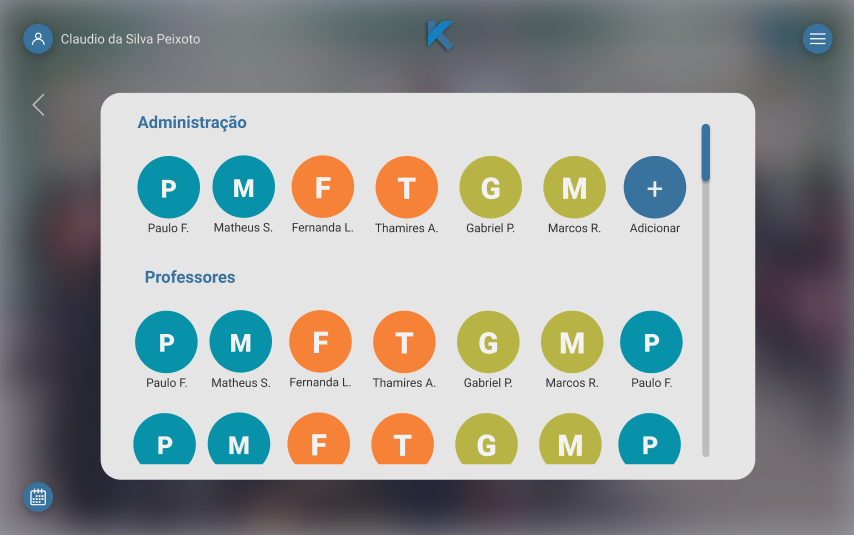


Figura 30 - Tela de docentes

Parte do código da tela de docentes:

import React, { useEffect, useState } from "react";  
import "./styles.css";  
  
import TeacherProfile from "../../../components/InfoModals/TeacherProfile";  
import ContentCard from "../../../components/ContentCard";  
  
function Teachers() {  
 const [active, setActive] = useState(false);  
 const [schoolId, setSchoolId] = useState('');  
 const [professors, setProfessors] = useState([]);  
 const history = useHistory();  
  
 useEffect(() => {  
 (async () => {  
 const search = window.location.search;  
 const params = new URLSearchParams(search);  
 const id = params.get('id');  
 if(!id || id === null) return history.push('/home');  
 setSchoolId(id);  
 const newProfessors = await api.get(`/schools/${id}/professors`);  
 setProfessors(newProfessors.data || []);  
 console.log(newProfessors.data);  
 })()  
 }, []);

TELA DE PERÍODOS

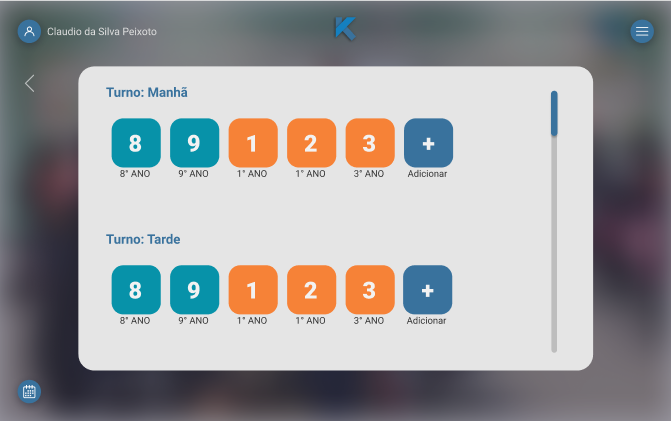


Figura 31 - Tela de períodos

Parte do código da tela de Períodos:

function handleClick(id:number) {  
 setSubjectId(id);  
 setActive(true);  
 }  
   
 return (  
 <div className="subjects-container">  
 <BackButton to={`/dashboard?id=${schoolId}`} />  
 <div className="scroll-view">  
 {groups.map(({name, module\_id}) => (  
 <div className="school-cards-container">  
 <h1>{name}</h1>  
 <div className="classes">  
 {subjects.map(({name, id, modules}:any, ind)=> {  
 return !modules.find((moduleData:any) => moduleData.module\_id === +module\_id)?null:(  
 <ContentCard  
 key={ind}  
 title={name.substr(0,3).toUpperCase()}  
 text={name}  
 color={getColor(name, ind)}  
 onClick={() => handleClick(id)}  
 style={{cursor: 'pointer'}}  
 />)  
 })}  
 <ContentCard  
 title="+"  
 text="Adicionar"  
 color={colors[18]}  
 onClick={() => {}}  
 style={{cursor: 'pointer'}}  
 />  
 </div>  
 </div>  
 ))}  
 </div>  
 <SubjectModal active={active} setActive={setActive} subjectId={subjectId} />  
 </div>  
 );

TELA DE RELATÓRIO

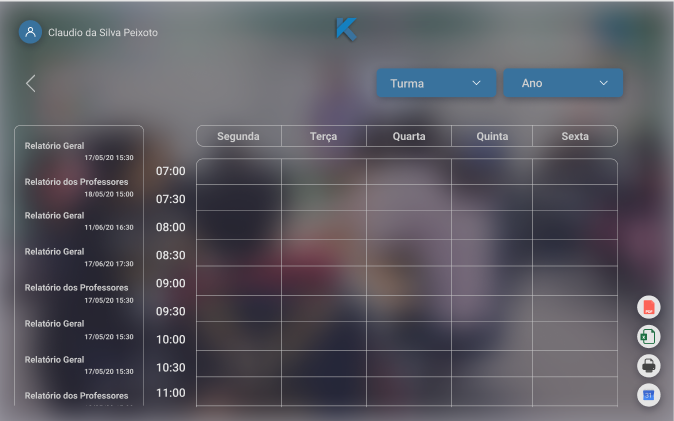


Figura 32 - Tela de relatório

Parte do código da tela de relatório:

import './styles.css';  
  
const Reports: React.FC = () => {  
 return (  
 <div className="main-reports">  
 <div className="reports-bar">  
 <ul>  
 <li>  
 <h1>Relatório Geral</h1>  
 <span>17/05/20 15:30</span>  
 </li>  
 <li>  
 <h1>Relatório Geral</h1>  
 <span>17/05/20 15:30</span>  
 </li>  
 <li>  
 <h1>Relatório Geral</h1>  
 <span>17/05/20 15:30</span>  
 </li>  
 </ul>  
 </div>  
 <Schedule sideContainerSize={400} to="/dashboard" />  
 </div>  
 );

PARTICIPAR DE UMA ESCOLA



Figura 33 - Tela para participar de uma escola

Parte do código da tela participação de uma escola:

return (  
 <CSSTransition  
 in={active}  
 timeout={300}  
 classNames="slide"  
 style={{  
 display: active ? "flex" : delayedActive ? "flex" : "none",  
 }}  
 >  
 <div className="global-school-menu">  
 <div className="school-menu-container">  
 <div  
 className="blur"  
 onClick={() => setActive(false)}  
 ></div>  
 <div className="school-menu-content">  
 <ul>  
 <li>  
 <a href="#create-school" onClick={()=> createSchool()} className="texts">  
 Criar uma nova escola  
 </a>  
 </li>  
 <li>  
 <a href="#enter-school" onClick={()=> enterSchool()} className="texts">  
 Entrar em uma nova escola  
 </a>  
 </li>  
 </ul>  
 </div>  
 </div>  
 </div>  
 </CSSTransition>  
 );  
};

# CRONOGRAMA

Para facilitar da visualização e organização do projeto dividimos o cronograma em três partes. A primeira se refere ao início do projeto, onde as ideias não estavam completamente formadas e foi necessário decidir os rumos do projeto, a problemática e fazer um panorama geral sobre tudo o que será realizado. Essa primeira etapa iniciou-se no dia 07/02/2020 e finalizou no dia 28/02 e suas tarefas estão descritas na Tabela 1 - Cronograma Etapa 1 a seguir.

Tabela 1 - Cronograma Etapa 1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ATIVIDADE** | **RESPONSÁVEL** | **PLANO DE INÍCIO** | **PLANO DE DURAÇÃO** | **INÍCIO REAL** | **DURAÇÃO REAL** |
| Definir os Integrantes do projeto | Todos | 07/02 | 7 | 07/02 | 3 |
| Definir as Responsabilidades | Todos | 07/02 | 6 | 07/02 | 6 |
| Escopo da Empresa | Todos | 07/02 | 4 | 07/02 | 7 |
| Definir o Impacto do Projeto | Todos | 11/02 | 8 | 11/02 | 6 |
| Decidir a Problemática | Todos | 07/02 | 11 | 17/02 | 12 |
| Viabilidade da Problemática | Todos | 11/02 | 3 | 16/02 | 6 |

Fonte: Do Autor (2020)

Logo após decidirmos todo o escopo do projeto e começarmos a planejar as suas características e desenvolvimentos de pesquisas, partimos para a etapa 2, onde planejamos como será o produto, realizamos protótipos e testes de ferramentas a serem utilizadas. Está etapa se iniciou no dia 01/05/2020 e finalizou no dia 29/06/2020. Segue as suas etapas e responsáveis na tabela 2.0 Cronograma etapa 2.

Tabela 2 - Cronograma Etapa 2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ATIVIDADE** | **RESPONSÁVEL** | **PLANO DE INÍCIO** | **PLANO DE DURAÇÃO** | **INÍCIO REAL** | **DURAÇÃO REAL** |
| Canvas e Persona | Todos | 01/05 | 7 | 01/05 | 7 |
| Funcionalidades | Todos | 07/05 | 7 | 07/05 | 21 |
| Jornada do Usuário | Todos | 07/05 | 7 | 07/05 | 3 |
| Requisitos do Sistema | Todos | 14/05 | 14 | 14/05 | 7 |
| Testes com o Ambiente GIT | Vinicius S. | 21/05 | 7 | 21/05 | 6 |
| Diagrama de Caso de Uso | Todos | 14/05 | 7 | 14/05 | 21 |
| Diagrama de Sequência | Matteo | 28/05 | 7 | 28/05 | 15 |
| Diagrama de Atividades | Matteo | 2/06 | 7 | 02/05 | 6 |
| Diagrama Entidade Relacionamento | Vitor | 24/05 | 7 | 28/05 | 10 |
| Testes com o Ambiente Python | Vinicius S. | 28/05 | 7 | 28/05 | 5 |
| Testes com o Ambiente Electron | Raphael | 09/05 | 3 | 09/05 | 3 |
| WireFrame das telas | Raphael | 14/05 | 15 | 11/06 | 7 |
| Preparação para Apresentação | Elmer | 02/06 | 7 | 09/06 | 7 |
| Rotas API | Vinicius S. | 15/06 | 14 | 15/06 | 14 |
| Logotipo | Raphael | 10/06 | 7 | 10/06 | 7 |
| Video de apresentação | Elmer | 10/06 | 7 | 14/06 | 7 |

Fonte: Do Autor (2020)

Após a segunda etapa partimos para a terceira que se divide em finalizar os protótipos e pesquisa e iniciar a fase da programação, colocando em prática diversos conceitos trabalhados. Ela decorre entre os dias de 05/07/2020 e 05/09/2020 e está demostrada com detalhes na Tabela 3 – Cronograma Etapa 3.

Tabela 3 - Cronograma Etapa 3

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ATIVIDADE** | **RESPONSÁVEL** | **PLANO DE INÍCIO** | **PLANO DE DURAÇÃO** | **INÍCIO REAL** | **DURAÇÃO REAL** |
|
| Protótipo Funcional | Vinicius P. | 05/07 | 14 | 05/07 | 14 |
| Banco de Dados | Vitor | 05/07 | 7 | 05/07 | 7 |
| Dicionário de Dados | Vitor | 12/07 | 7 | 12/07 | 7 |
| Diagrama de Classes | Vinicius P. | 05/07 | 7 | 05/07 | 7 |
| Fluxograma geração de relatório | Vinicius S. | 12/07 | 14 | 12/07 | 14 |
| Pesquisar as conexões de login | Vinicius S. | 26/07 | 7 | 26/07 | 7 |
| Rotas de usuário | Vitor | 03/08 | 7 | 03/08 | 7 |
| Rotas de escolas | Vinicius S. | 03/08 | 7 | 03/08 | 7 |
| Rotas de classes | Vitor | 15/08 | 7 | 15/08 | 7 |
| Rotas de professores | Vinicius S. | 21/08 | 7 | 21/08 | 7 |
| Rotas de turno | Vinicius S. | 01/09 | 5 | 01/09 | 5 |
| Tela de login | Raphael | 19/07 | 7 | 19/07 | 7 |
| Tela de início | Raphael | 26/07 | 7 | 26/07 | 7 |
| Telas do menu | Raphael | 03/08 | 7 | 03/08 | 7 |
| Tela do calendário | Raphael | 10/08 | 7 | 10/08 | 7 |
| Tela de perfil | Raphael | 17/08 | 7 | 17/08 | 7 |
| Tela de cadastro de escola | Raphael | 24/08 | 12 | 24/08 | 12 |
| Monografia - introdução, histórico, problema, hipóteses, objetivos gerais e específicos, justificativa, cronograma | Matteo | 05/07 | 7 | 05/07 | 7 |
| Monografia - entrevista com o especialista, softwares e equipamentos utilizados, explicação do programa | Matteo | 12/07 | 14 | 12/07 | 14 |
| Monografia - conclusão e revisão | Matteo | 26/07 | 7 | 26/07 | 7 |
| Entrevista para comentar as telas | Elmer | 05/07 | 14 | 05/07 | 14 |
| Video institucional (pitch) | Elmer | 19/07 | 14 | 19/07 | 14 |

Fonte: Do Autor (2020)

Quarta etapa do projeto, responsável pela finalização programação do projeto além da finalização da monografia e testes específicos.

Tabela 4 - Cronograma Etapa 4

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ATIVIDADE** | **RESPONSÁVEL** | **PLANO DE INÍCIO** | **PLANO DE DURAÇÃO** | **INÍCIO REAL** | **DURAÇÃO REAL** |
|
| Seeds do banco de dados | Vitor | 1 | 7 | 1 | 7 |
| Testes da API Web | Matteo | 1 | 14 | 1 | 14 |
| Rotas relatório professor | Vinicius S. | 15 | 7 | 15 | 7 |
| Rotas relatório classe | Vinicius S. | 15 | 7 | 15 | 7 |
| Hospedagem do banco e API | Vinicius P. | 22 | 7 | 22 | 7 |
| Icones para a criação da escola | Raphael | 1 | 14 | 1 | 14 |
| Responsividade da aplicação WEB | Vinicius P. | 1 | 7 | 1 | 7 |
| Animações da aplicação WEB | Vinicius P. | 8 | 7 | 8 | 7 |
| Conexão da aplicação WEB com o Backend | Raphael | 15 | 7 | 15 | 7 |
| Testes Frontend | Vinicius P. | 22 | 7 | 22 | 7 |
| Hospedagem do frontend | Vinicius P. | 22 | 7 | 22 | 7 |
| Realização do 2° vídeo | Todos | 8 | 14 | 8 | 14 |
| Finalização do diário de Bordo | Elmer | 22 | 7 | 22 | 7 |
| Finalização da documentação no GIT | Vinicius P. | 29 | 7 | 29 | 7 |
| Finalização da monografia | Elmer | 29 | 14 | 29 | 14 |
| Preparação para apresentação final | Todos | 29 | 7 | 29 | 7 |
| Teste do funcionamento total em escolas | Elmer | 36 | 21 | 36 | 21 |
| Seeds do banco de dados | Vitor | 1 | 7 | 1 | 7 |
| Testes da API Web | Matteo | 1 | 14 | 1 | 14 |
| Rotas relatório professor | Vinicius S. | 15 | 7 | 15 | 7 |

Fonte: Do Autor (2020)

# CONSIDERAÇÕES FINAIS

O sistema Klass tem como o objetivo pragmático trazer facilidade e tranquilidade a carga horária dos coordenadores de intuições acadêmicas que comumente realizar a criação das grades horárias, neste trabalho tivemos a demanda e o foco com o diretor da Faculdade Lusófona de São Paulo, Cesar Cavinato. Como objetivo principal do trabalho, a criação de um sistema que automatizasse o processo de criação da grade horária das aulas de acordo os horários dos professores. De acordo com nossas pesquisas entendemos que o excessivo acúmulo de trabalhos não só se dava ao com o Cesar, cliente do Projeto, mas que era um problema que ocorre em todos o meio acadêmico, se sabe que as instituições acadêmica são o centro de formação intelectual da sociedade, desta forma o sistema traria maior dinâmica a gestão acadêmica em consequência melhor produtividade, beneficiando não só o meio acadêmico, a sociedade como um todo.

No desenvolvimento o grupo enfrentou dificuldades em relação a agendamento de entrevistas com clientes e comunicação interna entre os participantes que logo se resolveu, em momentos específicos houveram problemas com o cronograma e em determinadas apresentações parciais do projeto mas que se resolveram com urgência, mas o projeto conseguiu todas informações, requisitos e recursos necessários para a realização do projeto não tendo limitações em relação a este tipo de problema, o que permitiu o bom andamento no decorrer do desenvolvimento do sistema.

# REFERÊNCIAS

React.Component. **React.** Disponível em: <https://pt-br.reactjs.org/docs/react-component.html>. Acesso em: 30 de set. de 2020.

Pensando do jeito React. **React.** Disponível em: <https://pt-br.reactjs.org/docs/thinking-in-react.html>. Acesso em: 30 de set. de 2020.

Solicitar Ciclo de Vida. Adonis. Disponível em: <https://adonisjs.com/docs/4.1/request-lifecycle>. Acesso em 30: de set. de 2020.

SANTOS, Carolina da Costa; PEREIRA, Fátima; LOPES, Amélia. Experiências da Gestão Acadêmica da Docência Universitária.**Educ. Real.**, Porto Alegre, v. 43, n. 3, p. 989-1008, Sept. 2018. Avaliado em <http://www.scielo.br/scielo.php?script =sci\_ arttext&pid=S2175-62362018000300989&lng=en&nrm=iso>. Acesso em 18 nov. 2020.  Epub Apr 09, 2018.  https://doi.org/10.1590/2175-623674106.

SOUZA, Maria Naires Alves de; MONTEIRO, André Jalles. Os docentes da Universidade Federal do Ceará e a utilização de alguns dos recursos do sistema integrado de gestão de atividades acadêmica (SIGAA). Ensaio: aval.pol.públ.Educ.,  Rio de Janeiro ,  v. 23, n. 88, p. 611-630,  Sept.  2015.   Available from < http://www.scielo.br/scielo.php?script=sci\_arttext&pid=S0104-40362015000300611&lng=en&nrm=i so>. Acesso em 18 nov.  2020.  https://doi.org/10.1590/S0104-40362015000300004.

Cordeiro Farias Filho, Doutor Milton. GESTÃO DE PROCESSO NA IMPLANTAÇÃO DE UM SISTEMA DE INFORMAÇÃO ACADÊMICA: A EXPERIÊNCIA DA UFPA. Florianópolis. Revista GUAL, v. 7, n.2, p.69-85, mai. 2014.

FONSECA, Marília; FONSECA, Dirce Mendes da. A gestão acadêmica da pós-graduação lato sensu: o papel do coordenador para a qualidade dos cursos.Educ. Pesqui., São Paulo, v. 42, n. 1, p. 151-164, Mar.  2016. Avaliado por <http://www.scielo.br/scielo.php?script=sci\_arttext&pid=S1517- 97022016000100151&lng=en&nrm= iso>. Acesso em 19 nov. 2020.  https://doi.org/10.1590/S1517-9702201603136263.