

CENTRO ESTADUAL DE EDUCAÇÃO TECNOLÓGICA PAULA SOUZA
Faculdade de Tecnologia de Jundiaí – “Deputado Ary Fossen”
Curso Superior de Tecnologia em Gestão da Tecnologia da Informação

Otávio Barros - 1140781923014
Victor Grossi Fróes - 1140781923020

DESENVOLVIMENTO DE APLICAÇÃO MOBILE – AGENDA ONLINE

Jundiaí
2022

Otávio Barros
Victor Grossi Fróes

DESENVOLVIMENTO DE APLICAÇÃO MOBILE – AGENDA ONLINE

Trabalho de Graduação apresentado à Faculdade de Tecnologia de Jundiaí - “Deputado Ary Fossen” como requisito parcial para a obtenção do título de Tecnólogo em Gestão da Tecnologia da Informação, sob a orientação da Professora Mestre Luciana Ferreira Baptista.

Jundiaí
2022

Faculdade de Tecnologia de Jundiaí - "Deputado Ary Fossen"
TCC-T2 - TERMO DE ACEITE DO PROFESSOR ORIENTADOR

Eu, Professor(a) Mestre Luciana Ferreira Baptista, docente do Curso de Gestão da Tecnologia da Informação da Faculdade de Tecnologia de Jundiaí - "Deputado Ary Fossen", declaro para os devidos fins que aceito a orientação do Trabalho de Conclusão de Curso que tem por tema principal: Agenda Mobile com rede de conexões (Notes.us) e que será elaborado pelo(s) estudante(s), nomeado(s) a seguir.

Nome Completo

Otavio Barros

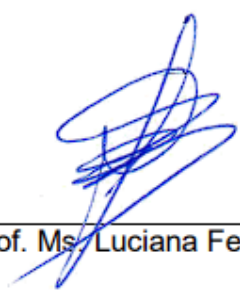
Victor Grossi Fróes

RA

1140781923014

1140781923020

Jundiaí, 04 de março de 2022



Prof. Ms. Luciana Ferreira Baptista

Dedicamos esse trabalho a nossos pais e amigos, que sempre estiveram do nosso lado nos momentos de dificuldade, e a todos os professores envolvidos em nossa formação, que sempre nos auxiliaram e transmitiram conhecimento.

AGRADECIMENTOS

Agradecemos aos nossos pais por sempre nos apoiarem e motivarem na busca de uma formação, e por nos ajudarem sempre nos momentos mais difíceis.

Ao corpo docente do Curso Superior de Tecnologia em Gestão da Tecnologia da Informação da Fatec Jundiaí, pelos ensinamentos que nos foram passados ao longo do curso, e em especial à nossa orientadora, Professora Luciana Baptista com todo o apoio em questões técnicas, e ao Professor Rafael Gross por toda a ajuda com as ideias de abordagem deste trabalho de conclusão de curso.

O aluno é como uma pequena semente que deve ser plantada e cuidada para germinar e dar bons frutos. O professor é como o agricultor que vê na semente a esperança que proverá as necessidades da sociedade.

Luis Alves

BARROS, Otavio e Fróes, Victor Grossi. **DESENVOLVIMENTO DE APLICAÇÃO MOBILE – AGENDA ONLINE**. 42 f. Trabalho de Conclusão de Curso de Tecnólogo em Gestão da Tecnologia da Informação. Faculdade de Tecnologia de Jundiaí - “Deputado Ary Fossen”. Centro Estadual de Educação Tecnológica Paula Souza. Jundiaí. 2022.

RESUMO

Agenda Connect é um aplicativo mobile que permite que você veja os eventos de seus amigos em um só lugar. Ele permite que você compartilhe facilmente os eventos com outros e torna mais fácil ver o que seus amigos planejaram. A plataforma atende a todas as necessidades e exigências de um indivíduo ocupado. Ela é feita com programação de front-end em React Native junto com JavaScript e back-end feito em ASP.NET Core e C#. Atualmente, as pessoas utilizam ferramentas como Google Agenda, Teams ou Calendar para coordenar os horários de todos os envolvidos em um evento. Entretanto, para que as pessoas possam ver os calendários uns dos outros, é necessário que haja um local central onde todas essas informações sejam armazenadas, são inúmeros menus que o usuário precisa aprender, daí surge a ideia da Agenda Connect, em poucos menus e com tempo de uso de 3 minutos a pessoa já consegue entender todo o funcionamento do aplicativo.

Palavras-chave: Agenda. Front-end. React Native. Back-end. Asp.Net Core 5

BARROS, Otavio e Fróes, Victor Grossi. **DESENVOLVIMENTO DE APLICAÇÃO MOBILE – AGENDA ONLINE**. 42 f. Trabalho de Conclusão de Curso de Tecnólogo em Gestão da Tecnologia da Informação. Faculdade de Tecnologia de Jundiaí - “Deputado Ary Fossen”. Centro Estadual de Educação Tecnológica Paula Souza. Jundiaí. 2022.

ABSTRACT

Agenda Connect is a mobile app that lets you see your friends' events in one place. It allows you to easily share events with others and makes it easy to see what your friends have planned. The platform caters to all the needs and requirements of a busy individual. It is made with front-end programming in React Native along with JavaScript and back-end made in ASP.NET Core and C#. Currently, as people use tools like Google Calendar, Teams or Calendar to coordinate the schedules of everyone involved in an event. However, to see each other's calendars, it is necessary that the information is stored in a central location, where all users need to learn, from all menus and agenda with Connect, in a use time of 3 minutes the person can already understand the entire operation of the application.

Keywords: Agenda. Front-end. React Native. Back-end. Asp.Net Core 5

LISTA DE ILUSTRAÇÕES

Figura 1 – Primeiro caso de uso: Módulo Usuário.....	20
Figura 2 – Segundo caso de uso: Módulo Agendas.....	20
Figura 3 – Terceiro caso de uso: Módulo Conexões.....	21
Figura 4 – Interface IQueriesService.....	26
Figura 5 – Classe QueriesService.....	27
Figura 6 – Classe UsuarioController.....	27
Figura 7 – Classe AgendaController.....	28
Figura 8 – Classe ConexaoController.....	28
Figura 9 – Diagrama de Entidade Relacionamento do banco de dados.....	29
Figura 10 – Tela de login do sistema.....	36
Figura 11 – Autorização para a sessão com o Expo.....	37
Figura 12 – Tela de conexões.....	38
Figura 13 – Tela de agendas.....	39
Figura 14 – Tela de calendário com todas as agendas.....	40
Figura 15 – Tela com solicitações de conexões.....	41
Figura 16 – Tela com informações do usuário.....	42

LISTA DE TABELAS

Tabela 1 – Requisitos Funcionais.	17
Tabela 2 – Requisitos Não Funcionais.....	18
Tabela 3 – Descrição do caso de uso “Criar Usuário”	21
Tabela 4 – Descrição do caso de uso “Login Google”.....	22
Tabela 5 – Descrição do caso de uso “Buscar Usuário”.	22
Tabela 6 – Descrição do caso de uso “Solicitar Agenda”.....	22
Tabela 7 – Descrição do caso de uso “Criar Agenda”.....	23
Tabela 8 – Descrição do caso de uso “Deletar Agenda”.	23
Tabela 9 – Descrição do caso de uso “Atualizar Agenda”.....	23
Tabela 10 – Descrição do caso de uso “Solicitar Conexão”.....	24
Tabela 11 – Descrição do caso de uso “Aceitar Conexão”.	24
Tabela 12 – Descrição do caso de uso “Recusar Conexão”.	24
Tabela 13 – Descrição do caso de uso “Excluir Conexão”.....	25
Tabela 14 – Descrição do caso de uso “Visualizar Conexão”.	25
Tabela 15 – Dicionário de Dados da tabela USUARIOS.....	30
Tabela 16 – Dicionário de Dados da tabela AGENDAS.....	30
Tabela 17 – Dicionário de Dados da tabela CONEXOES.	31

SUMÁRIO

1	INTRODUÇÃO	12
2	ESPECIFICAÇÃO DO PROGRAMA	15
2.1	Escopo.....	15
2.2	Clientes do software	15
3	REQUISITOS DO SISTEMA.....	17
3.1	Requisitos Funcionais	17
3.2	Requisitos não funcionais.....	18
4	DEFINIÇÃO DO PROJETO	19
4.1	Casos de Uso.....	20
4.2	Diagramas de Classe	25
4.3	Banco de Dados	29
4.4	Diagrama Entidade Relacionamento	29
4.5	Dicionário de Dados.....	29
5	NOTAS DE ATUALIZAÇÕES FUTURAS	32
6	CONSIDERAÇÕES FINAIS.....	34
	REFERÊNCIAS.....	35
	APÊNDICE A - MANUAL DO USUÁRIO	36

1 INTRODUÇÃO

A aplicação é uma agenda mobile com objetivo de conexão de pessoas e suas agendas, que oferece aos usuários um espaço para administrar suas tarefas em uma interface simples.

Existem diversas aplicações de gerenciamento de tarefas mantidas por grandes empresas, e o objetivo desta não é competir com tais, mas sim oferecer uma opção com interface simples e com recursos simplificados, atendendo qualquer que seja a demanda dos usuários.

A aplicação será desenvolvida no ambiente Mobile, usando no Back-End uma API desenvolvida em .NET 5, e no Front-End (Interface do usuário & Interações) o framework React Native, utilizando a ferramenta EXPO que facilita a criação de aplicativos Mobile, para agregar toda a lógica do Front, com auxílio do React Native Elements, uma biblioteca para interface do usuário mantida pelo Facebook, oferecendo diversos recursos gráficos que já fazem parte do cotidiano de o usuário comum nos Apps Mobile.

A ideia do App nasceu com o advento da Pandemia da COVID-19, que ainda é algo bastante recorrente. Devido a este fato, notamos a necessidade de unir pessoas mesmo que elas não estivessem presentes, era demasiadamente necessário que as pessoas estivessem com suas agendas em dia, mesmo que não se trata-se de um ambiente corporativo, daí nasce a ideia do Agenda Connect uma plataforma que pudesse unir as agendas de pessoas que estão conectadas mesmo sem estar perto. A população que tem seu emprego e ao mesmo tempo estuda no ensino superior, muitas vezes encontra-se obrigada a administrar seus afazeres, por meio de aplicações do tipo “To Do” / “Agenda Online”. Ferramentas como Notion e Slack, que são mantidas por grandes empresas, contendo diversas funcionalidades, são ótimas opções de “agendas online”. Porém o fenômeno do aumento da utilização dos ambientes virtuais, somado às várias opções de customização dessas aplicações, abrem brechas para dificuldades no “user-side”, provenientes, por exemplo, do cansaço gerado pelas longas horas em frente ao computador.

Uma aplicação mais simples e com menos recursos pode oferecer uma solução que apesar de simples, tenha uma abordagem prática. Da sua interface até a maneira de processar dados, seu desenvolvimento seria feito pensando em acessibilidade e

desempenho. Ou seja, uma aplicação com uma interface do usuário de fácil uso, aliado de uma arquitetura de dados otimizada, somado a poucas funcionalidades, preencheria esta lacuna de mercado.

A proposta não é concorrer e sim auxiliar as aplicações já consolidadas no mercado, oferecendo aos usuários, um espaço simplificado para anotações de tarefas, que contenham mais informações que um despertador (Hora & Título), porém menos informações do que o escopo completo para o desenvolvimento de um projeto (Cliente, Prazos, Público-alvo, Orçamento, Etapas, etc.). O espaço em que a aplicação se encaixa, serve como um poderoso ajudante das demais agendas que o usuário possua. Uma vez em que ela já é projetada para não armazenar grandes volumes de dados por tarefa.

Tem por objetivo geral cobrir a lacuna do mercado, de uma agenda online comum, que forneça a possibilidade de gerir tarefas sem necessidade de planos pagos, propagandas ou interfaces confusas, e por específico criar uma interface de fácil interação, mas que cumpra o seu propósito, anotar tarefas pendentes, oferecendo uma alternativa em meio aos concorrentes de big techs.

A metodologia de pesquisa escolhida foi quantitativa. A decisão foi tomada ao perceber que para desenvolver uma solução simples que cumpra as necessidades do usuário muito atribulado ou dos mais idosos adentrando o mundo virtual, seria necessário encontrar um ponto de equilíbrio entre esses tipos de usuário. Tanto na escolha das tecnologias que influenciarão diretamente o desempenho do software quanto na escolha das cores da interface, tudo deveria encontrar um equilíbrio para que não se tornasse uma agenda online comum e(ou) confusa, ainda assim com o foco de conexão de usuários, visto que a agenda se tem um viés de rede social. Esse método de pesquisa utiliza os dados obtidos nas respostas e os traduz em números, que mostram a reação do usuário em relação ao seu produto, que no caso deste trabalho, é o software a ser desenvolvido (QUALIBEST, 2020).

Porém por não possuir experiência no desenvolvimento de um sistema desde seu início, a dupla precisará realizar pesquisas de caráter experimental, para mapear possíveis falhas de interpretação e desenvolvimento, sabendo todas as variantes possíveis é mais fácil de identificar a melhor maneira de aplicação. Exemplo prático desta pesquisa seria com o protótipo da aplicação consideravelmente avançando,

requisitar o teste por parte de terceiros que se encaixem nos padrões desejados e não estejam diretamente ligados com o desenvolvimento do software.

2 ESPECIFICAÇÃO DO PROGRAMA

2.1 Escopo

O aplicativo tem como objetivo principal a conexão de agenda de pessoas, ou seja, não é voltado para empresas. Funcionará da seguinte forma:

1. Na primeira tela do sistema, o usuário faz Login via Google, ou seja, não é necessário se preocupar com cadastro de informações pessoais dentro da aplicação.
2. Uma vez logado, o usuário é redirecionado para a segunda tela com sua lista de conexões.
3. Na terceira tela é possível ver todas as suas agendas.
4. Na quarta tela é possível ver um calendário e todas as agendas, suas e de suas conexões.
5. Na quinta tela é possível ver as suas solicitações de conexões.
6. E, por último, na sexta tela é possível ver suas informações pessoais e seu código de conexão, que você pode mandar para seus amigos se conectarem, utilizando este código juntamente com seu e-mail.

O aplicativo tem poucas funcionalidades, seu propósito não é gerar cadeias de conexões e agendas de reuniões, mas simplesmente ver em quais momentos seus amigos estão atarefados e vice-versa. Desta maneira, não é possível vincular pessoas além de você em uma agenda.

2.2 Clientes do software

O cliente principal do Software são as pessoas físicas, a ideia da aplicação como citado anteriormente não é gerar agendas para organizações corporativas envolverem seus funcionários, mas sim para aquelas pessoas que desejam visualizar quais são as tarefas atuais de seus colegas e suas, e utilizarem a ferramenta de maneira produtiva para organizar viagens, passeios, ou comunhões pessoais.

O aplicativo por ser bastante simples de utilizar é direcionado para todos os públicos, mas pelos estudos que fizemos temos a ideia de que o público que mais irá

utilizar a aplicação está dentro da faixa etária dos 18 aos 47 anos. Os dados foram levantados a partir de pesquisas mencionadas ao público alvo visto no próprio site do Google.

3 REQUISITOS DO SISTEMA

Os Requisitos Funcionais são as necessidades ou solicitações que o software deverá realizar. Vale lembrar que vários Requisitos Funcionais podem ser executados dentro de uma mesma funcionalidade do sistema (CANGUÇU, 2021).

Podemos dizer que os Requisitos Não Funcionais são como o sistema executará os Requisitos Funcionais, ou seja, tudo relacionado ao hardware, software, sistema operacional, etc. (CANGUÇU, 2021).

3.1 Requisitos Funcionais

Os Requisitos Funcionais que farão parte do sistema serão os seguintes:

#	Requisitos Funcionais	Casos de uso
RF01	O sistema deverá criar o usuário.	UC01.01
RF02	O sistema deverá possibilitar o login com o Google.	UC01.02
RF03	O sistema deverá permitir a busca de usuários.	UC01.03
RF04	O sistema deverá permitir a exibição das agendas cadastradas.	UC02.01
RF05	O sistema deverá permitir a criação de agendas.	UC02.02
RF06	O sistema deverá possibilitar a exclusão das agendas cadastradas.	UC02.03
RF07	O sistema deverá possibilitar a atualização das agendas cadastradas.	UC02.04
RF08	O sistema deverá possibilitar ao usuário a solicitação de conexão com outras pessoas.	UC03.01
RF09	O sistema deverá possibilitar ao usuário aceitar conexão com outras pessoas.	UC03.02
RF10	O sistema deverá possibilitar ao usuário recusar conexão com outras pessoas.	UC03.03
RF11	O sistema deverá possibilitar ao usuário excluir conexão com outras pessoas.	UC03.04
RF12	O sistema deverá possibilitar ao usuário a exibição das conexões aceitas e em aberto.	UC03.05

Tabela 1 – Requisitos Funcionais.

3.2 Requisitos não funcionais

Os Requisitos Não Funcionais que serão necessários para a execução do software serão os seguintes:

#	Requisitos não funcionais
RNF01	O Banco de dados utilizado deverá ser o MySQL.
RNF02	Será implementada uma API em .NET Core 5 para o Back-End.
RNF03	O Front-end será implementado em React Native.
RNF04	O sistema utilizará o Firebase para receber as informações de login do usuário.
RNF05	Será necessário um dispositivo móvel com acesso à internet para utilizar o sistema.

Tabela 2 – Requisitos Não Funcionais.

4 DEFINIÇÃO DO PROJETO

A aplicação foi desenvolvida com base na Programação Orientada a Objetos, que é um padrão de desenvolvimento de diversas linguagens, e se baseia em quatro pilares (DEVMEDIA, 2014):

1. Abstração: Um dos pontos mais importantes no paradigma da Programação Orientada a Objetos, o objeto que é tratado dentro do sistema é a representação de um objeto real, então é preciso imaginar o que ele irá fazer dentro do sistema. É necessário ter uma identidade, propriedades e métodos.
2. Encapsulamento: Adiciona segurança na aplicação, pois esconde as propriedades e define seus valores apenas onde é desejado que seja feito, evitando que fique exposto em todo o restante da aplicação.
3. Herança: Caracteriza o reaproveitamento de código na Programação Orientada a Objetos, na qual objetos e classes literalmente herdam propriedades e métodos vindos de outros objetos e classes.
4. Polimorfismo: Em alguns casos, o mesmo método precisa ter uma funcionalidade diferente quando é herdado de sua classe pai. Essa é a função do polimorfismo, que é uma característica extremamente ligada à herança.

Escolhemos desenvolver o sistema baseado na Orientação a Objetos, pois possuímos experiência de trabalho com esse paradigma, e além disso, grande parte das aplicações atualmente são feitas nesse modelo, sendo assim a melhor estrutura para criação de um projeto atualmente.

O banco de dados escolhido foi o MySQL, devido a sua facilidade e flexibilidade de uso, e também a seu alto desempenho que garante consultas e respostas rápidas. O MySQL é um Banco de Dados que segue o modelo relacional, ou seja, todos os seus dados são armazenados e arrumados em tabelas (LONGEN, 2021).

4.1 Casos de Uso

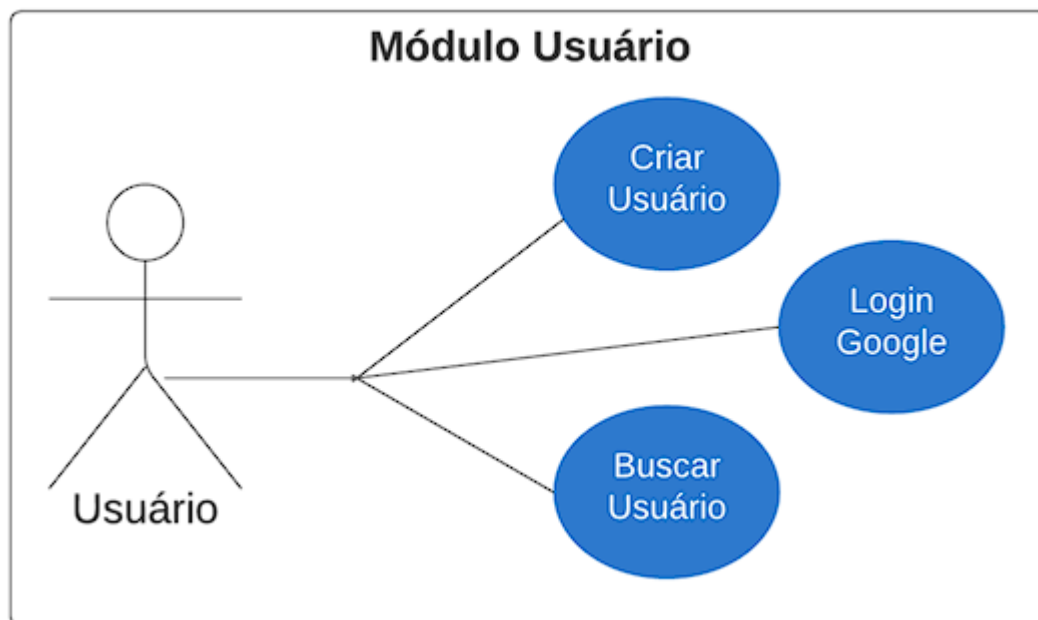


Figura 1 – Primeiro caso de uso: Módulo Usuário.

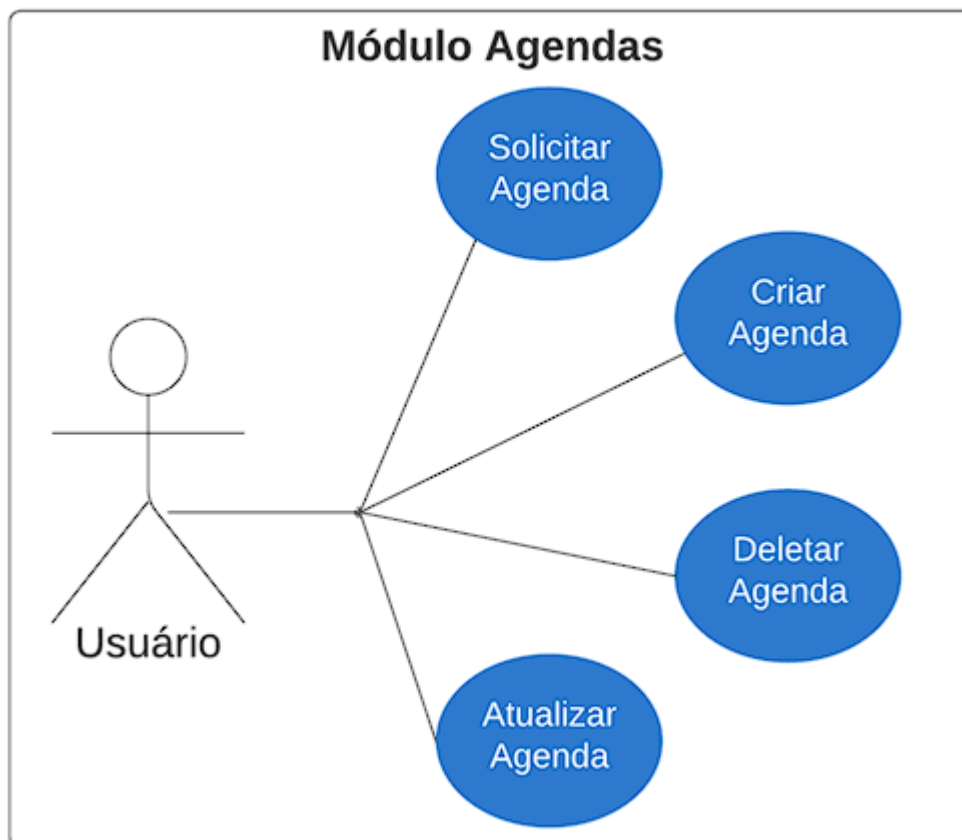


Figura 2 – Segundo caso de uso: Módulo Agendas.

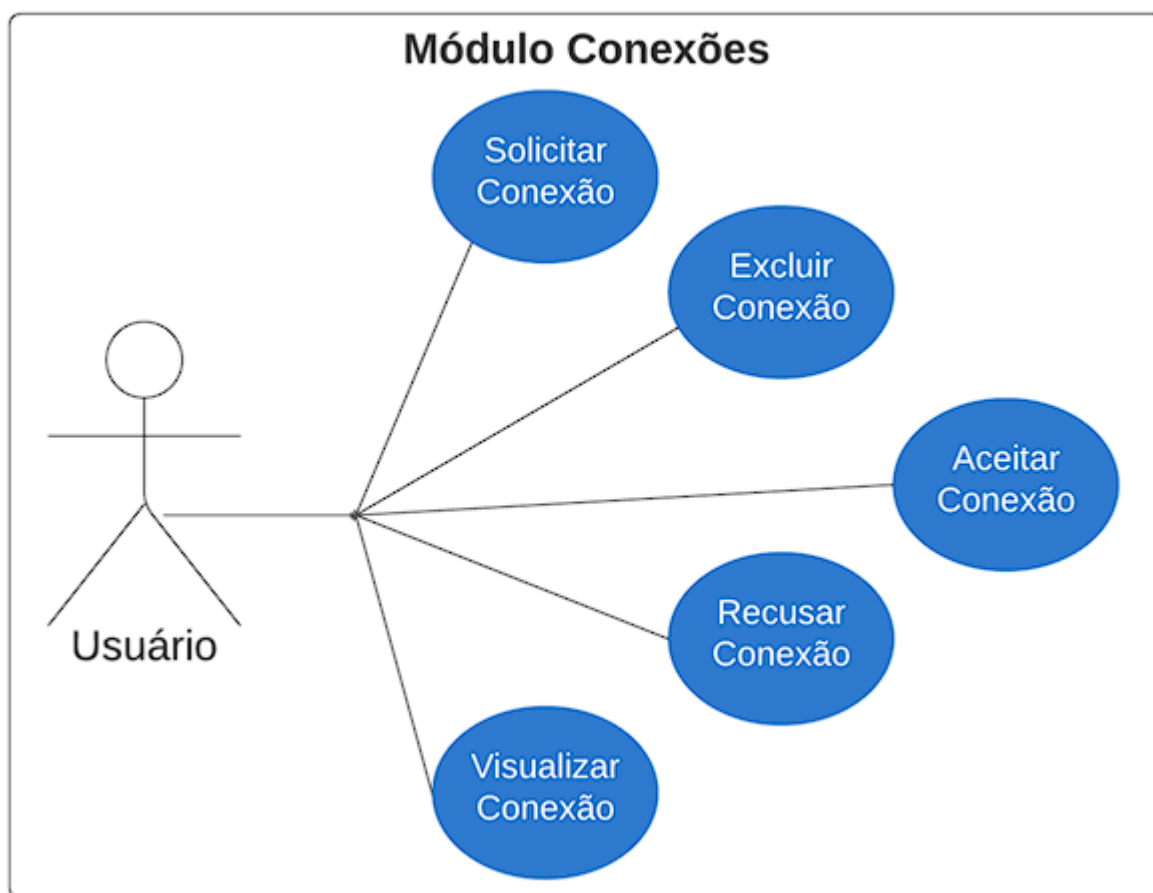


Figura 3 – Terceiro caso de uso: Módulo Conexões.

Nas tabelas a seguir são abordados os usos de caso da aplicação. Será possível visualizar que existe apenas um usuário final, já que a aplicação não contém parte administrativa, desta maneira, foi montado o Sistema da seguinte maneira:

Nome	Criar Usuário - UC01.01
Descrição	Criação de um novo usuário na plataforma
Atores	Usuário
Pré-Condição	Usuário conecta-se pela primeira vez à aplicação
Pós-Condição	(Sucesso) Criação de Conta e cadastro de dados (Falha) Não ocorre
Fluxo Principal	<ol style="list-style-type: none"> 1. Usuário entra na aplicação 2. Dados do usuário são carregados da resposta do Google Firebase Authentication 3. Dados retornados são salvos no banco de dados
Fluxo Alternativo	Não existe

Tabela 3 – Descrição do caso de uso "Criar Usuário".

Nome	Login Google – UC01.02
Descrição	Usuário faz autenticação via Google Firebase Auth
Atores	Usuário
Pré-Condição	Usuário “Starta” aplicação no dispositivo
Pós-Condição	Usuário é levado a página de conexões
Fluxo Principal	<ol style="list-style-type: none"> 1. Usuário “Starta” aplicação 2. Usuário é direcionado para uma tela onde é pedido login via Google 3. Usuário preenche informações conta Google 4. Usuário retornado para a tela de conexões
Fluxo Alternativo	Não existe

Tabela 4 – Descrição do caso de uso “Login Google”.

Nome	Buscar Usuário – UC01.03
Descrição	Busca-se um usuário fornecendo seu e-mail dentro da API
Atores	Usuário
Pré-Condição	Usuário estar autenticado.
Pós-Condição	<p>(Sucesso) Informações do usuário e de suas conexões são retornados</p> <p>(Falha) Caso o usuário não tenha conexões sua tela de conexões apresentará layout diferenciado.</p>
Fluxo Principal	<ol style="list-style-type: none"> 1. Usuário entra na aplicação 2. Executa-se uma query que busca as conexões do usuário 3. Caso a lista exista, retorna-se além dos dados do usuário os dados da lista de conexões
Fluxo Alternativo	Não existe

Tabela 5 – Descrição do caso de uso “Buscar Usuário”.

Nome	Solicitar Agenda – UC02.01
Descrição	Fornecendo um identificado, retorna-se todas as agendas de um usuário
Atores	Usuário
Pré-Condição	Usuário estar conectado na aplicação / Usuário ter conexões
Pós-Condição	<p>(Sucesso) Uma lista de agendas é retornada com base em identificadores fornecidos</p> <p>(Falha) Caso o usuário não tenha nenhuma agenda ou conexão com agendas, uma tela diferenciada é mostrada</p>
Fluxo Principal	<ol style="list-style-type: none"> 1. Id fornecido a API 2. Retorno com agendas baseado no Id fornecido
Fluxo Alternativo	Caso não seja encontrado não haverá renderização de agendas em tela

Tabela 6 – Descrição do caso de uso “Solicitar Agenda”.

Nome	Criar Agenda – UC02.02
Descrição	Cria-se uma agenda no banco de dados
Atores	Usuário
Pré-Condição	Estar conectado na aplicação, acessar o menu agendas
Pós-Condição	(Sucesso) Criação de agenda e cadastro de dados (Falha) Mensagem de erro retornada
Fluxo Principal	<ol style="list-style-type: none"> 1. Usuário entra no menu de criação de agendas 2. Escolhe data, título e descrição, e confirma a criação
Fluxo Alternativo	Caso o usuário não queira criar uma agenda ele clica em cancelar

Tabela 7 – Descrição do caso de uso “Criar Agenda”.

Nome	Deletar Agenda – UC02.03
Descrição	Deleta uma agenda baseada no ID fornecido
Atores	Usuário
Pré-Condição	Usuário possui agendas. Usuário é dono da agenda
Pós-Condição	(Sucesso) Agenda deletada da agenda (Falha) Mensagem de erro caso operação não seja concluída
Fluxo Principal	<ol style="list-style-type: none"> 1. Usuário acessa uma agenda já criada 2. Clica no botão que visualmente remete a deletar uma agenda 3. Mensagem de confirmação 4. Agenda deletada
Fluxo Alternativo	Não existe

Tabela 8 – Descrição do caso de uso “Deletar Agenda”.

Nome	Atualizar Agenda – UC02.04
Descrição	É possível atualizar informações de uma agenda
Atores	Usuário
Pré-Condição	Usuário possui agendas. Usuário é dono da agenda
Pós-Condição	(Sucesso) Dados atualizados e já visíveis em tela (Falha) Caso não seja possível atualizar a agenda, uma mensagem de erro é mostrada
Fluxo Principal	<ol style="list-style-type: none"> 1. Usuário clica em uma agenda já criada. 2. Usuário digita novos dados para a agenda. 3. Ao salvar a agenda é alterada
Fluxo Alternativo	Não existe

Tabela 9 – Descrição do caso de uso “Atualizar Agenda”.

Nome	Solicitar Conexão – UC03.01
Descrição	Usuário solicita conexões com outro usuário
Atores	Usuário
Pré-Condição	Usuário sabe o código de conexão e o e-mail da outra conexão
Pós-Condição	(Sucesso) Solicitação de conexão enviada com sucesso, aguarda resposta (Falha) Caso o usuário não seja encontrado, ou não seja possível adicioná-lo é mostrado uma mensagem
Fluxo Principal	<ol style="list-style-type: none"> 1. Usuário acessa o menu de solicitação de novas conexões 2. Usuário preenche id de conexão e e-mail de conexão. 3. Conexão solicitada
Fluxo Alternativo	Não existe

Tabela 10 – Descrição do caso de uso “Solicitar Conexão”.

Nome	Aceitar Conexão – UC03.02
Descrição	Usuário entra na sua lista de conexões solicitadas e aprova uma conexão
Atores	Usuário
Pré-Condição	Usuário foi solicitado a se conectar com outra pessoa e não a recusou previamente
Pós-Condição	(Sucesso) Criada uma conexão (Falha) Não ocorre
Fluxo Principal	<ol style="list-style-type: none"> 1. Usuário acessa seu menu de solicitações 2. Usuário acessa uma solicitação 3. Usuário aceita a conexão
Fluxo Alternativo	Não existe

Tabela 11 – Descrição do caso de uso “Aceitar Conexão”.

Nome	Recusar Conexão – UC03.03
Descrição	Usuário opta por não aceitar conexão, neste caso, o solicitante não poderá mais tentar adicioná-lo
Atores	Usuário
Pré-Condição	Usuário tem uma solicitação de conexão ativa.
Pós-Condição	(Sucesso) Solicitação recusada (Falha) Não ocorre
Fluxo Principal	<ol style="list-style-type: none"> 1. Usuário acessa seu menu de solicitações 2. Usuário acessa uma solicitação 3. Usuário aceita a conexão
Fluxo Alternativo	Não existe

Tabela 12 – Descrição do caso de uso “Recusar Conexão”.

Nome	Excluir Conexão – UC03.04
Descrição	Usuário exclui uma conexão já conectada previamente
Atores	Usuário
Pré-Condição	Usuário já é conectado a este outro usuário
Pós-Condição	(Sucesso) A conexão não existe mais, mas pode ser solicitada novamente. (Falha) Não ocorre
Fluxo Principal	<ol style="list-style-type: none"> 1. Usuário acessa seu menu de conexões 2. Usuário seleciona uma conexão 3. Usuário exclui a solicitação clicando no botão equivalente visual a exclusão e confirma operação
Fluxo Alternativo	Não existe

Tabela 13 – Descrição do caso de uso “Excluir Conexão”.

Nome	Visualizar Conexão – UC03.05
Descrição	Usuário visualiza as conexões em aberto e/ou aceitas
Atores	Usuário
Pré-Condição	Usuário possui solicitação de conexões em aberto e/ou conexões aceitas.
Pós-Condição	(Sucesso) Informações sobre as conexões são retornadas (Falha) Não ocorre
Fluxo Principal	<ol style="list-style-type: none"> 1. Usuário acessa seu menu de conexões 2. Usuário visualiza as conexões em aberto e as conexões aceitas
Fluxo Alternativo	Não existe

Tabela 14 – Descrição do caso de uso “Visualizar Conexão”.

4.2 Diagramas de Classe

Os Diagramas de Classe estão entre os principais tipos de diagramas que existem, com eles é possível mapear a estrutura do sistema modelando suas classes, atributos, operações e relações entre os objetos (LUCIDCHART, 2018). A representação das classes é feita por meio de um retângulo, dividido em três partes: a superior com o nome da classe, a do meio com os atributos, e a inferior com os métodos que compõem a classe. E a representação de interfaces é feita com um retângulo dividido em duas partes: a superior com o nome da interface, e a inferior com as definições dos métodos (UFCG, 2010).

O sistema tem uma interface, *IQueriesService*, com as definições dos métodos que serão usados para fazer as queries no banco de dados, conforme mostrado na Figura 4.

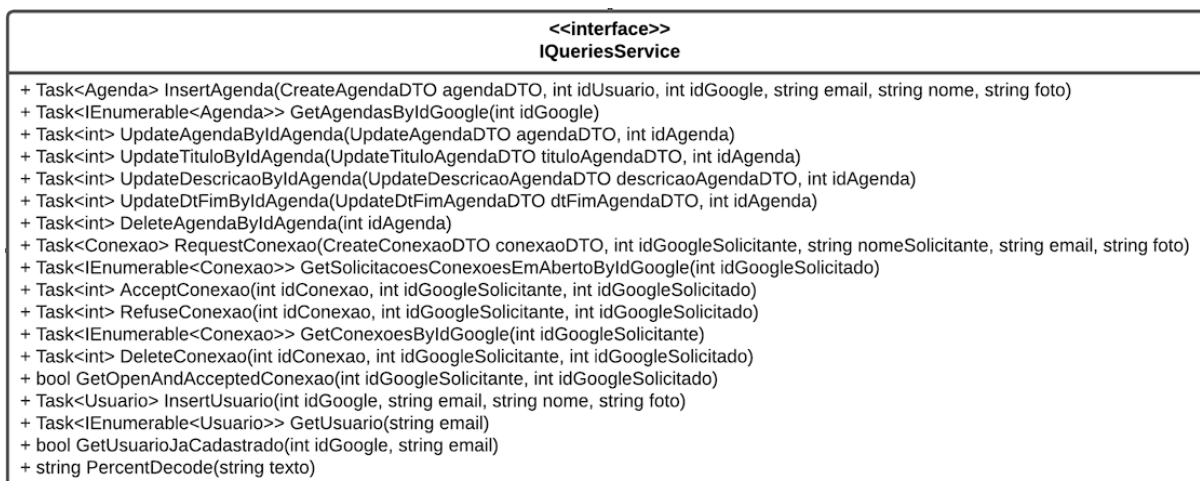


Figura 4 – Interface IQueriesService.

A classe *QueriesService*, mostrada na Figura 5, herda os métodos que foram implementados na interface *IQueriesService* e faz as queries que serão utilizadas posteriormente nas *controllers*, quando forem enviadas requisições para a API.

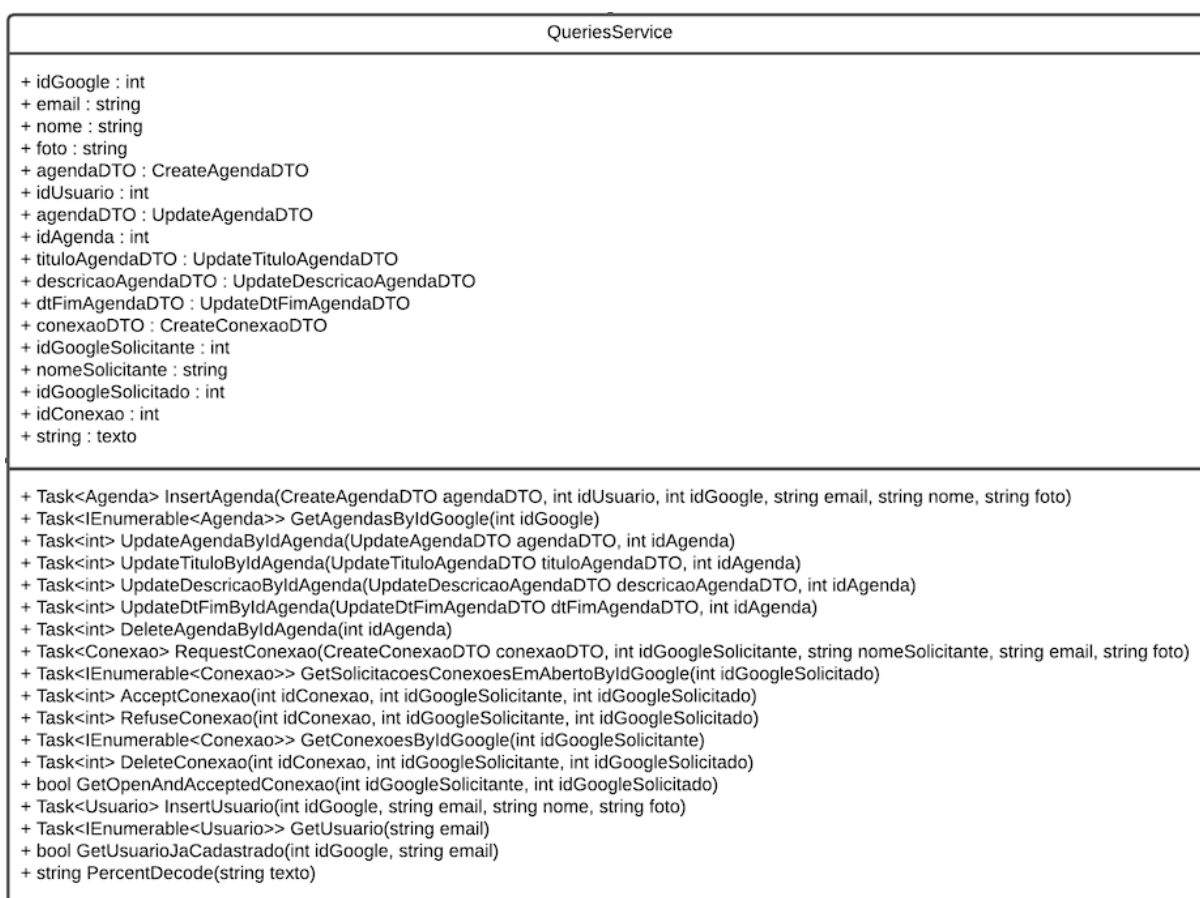


Figura 5 – Classe QueriesService.

A classe *UsuarioController*, mostrada na Figura 6, utiliza os métodos implementados na classe *QueriesService* que fazem as requisições referentes aos usuários, que são: cadastro de usuário e pesquisa de usuário.

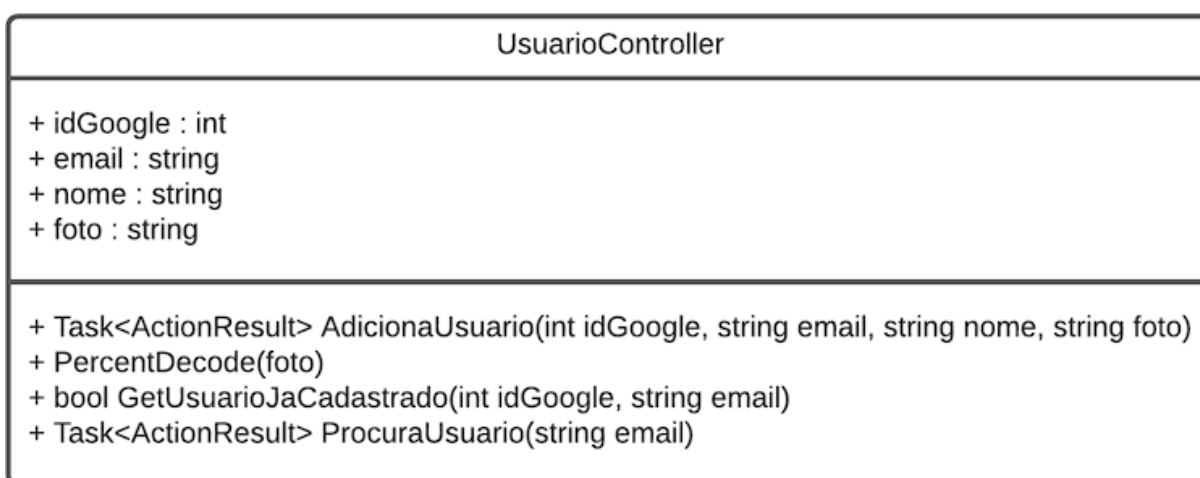


Figura 6 – Classe UsuarioController.

A classe *AgendaController* utiliza os métodos implementados na classe *QueriesService* que fazem as requisições referentes às agendas, que são: cadastro de agenda, exibição de agenda, alteração de agenda e exclusão de agenda, conforme mostrado na Figura 7.

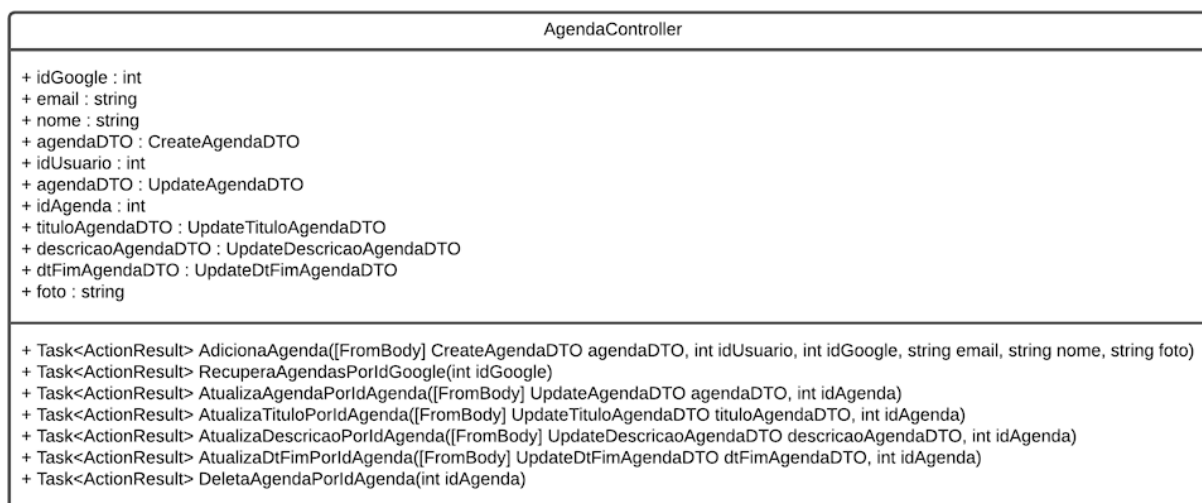


Figura 7 – Classe AgendaController.

Por último, temos a classe *ConexaoController* que utiliza os métodos implementados na classe *QueriesService* que fazem as requisições referentes às conexões entre pessoas, que são: solicitação de conexão com outras pessoas, exibição das solicitações de conexão em aberto, possibilidade de aceitar ou recusar conexões com outras pessoas, possibilidade de excluir conexões com outras pessoas e exibição das conexões aceitas, conforme mostrado na Figura 8.

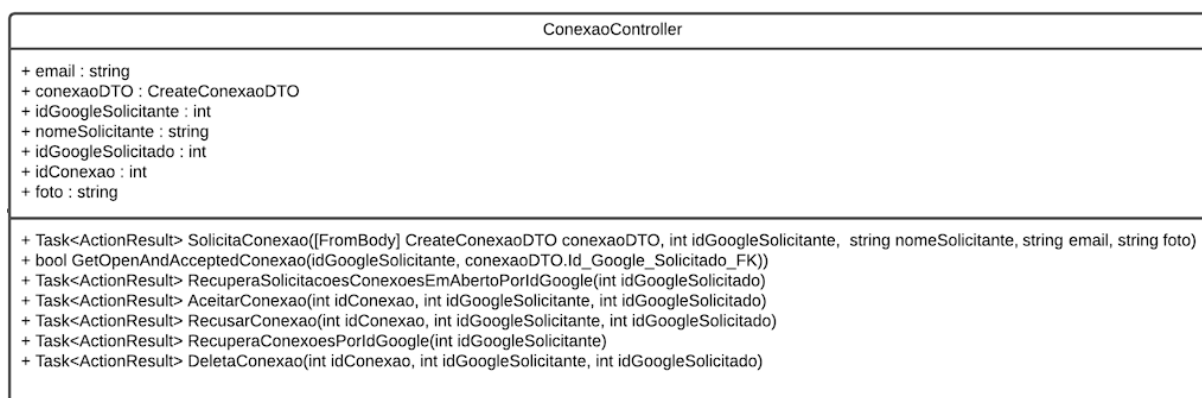


Figura 8 – Classe ConexaoController.

4.3 Banco de Dados

O Banco de Dados é composto por três tabelas: USUARIOS, AGENDAS e CONEXOES. Os dados do usuário são armazenados na tabela USUARIOS e seus campos chaves, ID_USUARIO, ID_GOOGLE, EMAIL e NOME são utilizados pelas outras tabelas para armazenar os dados das agendas e conexões entre os usuários, conforme mostrado na Figura 9 abaixo.

4.4 Diagrama Entidade Relacionamento

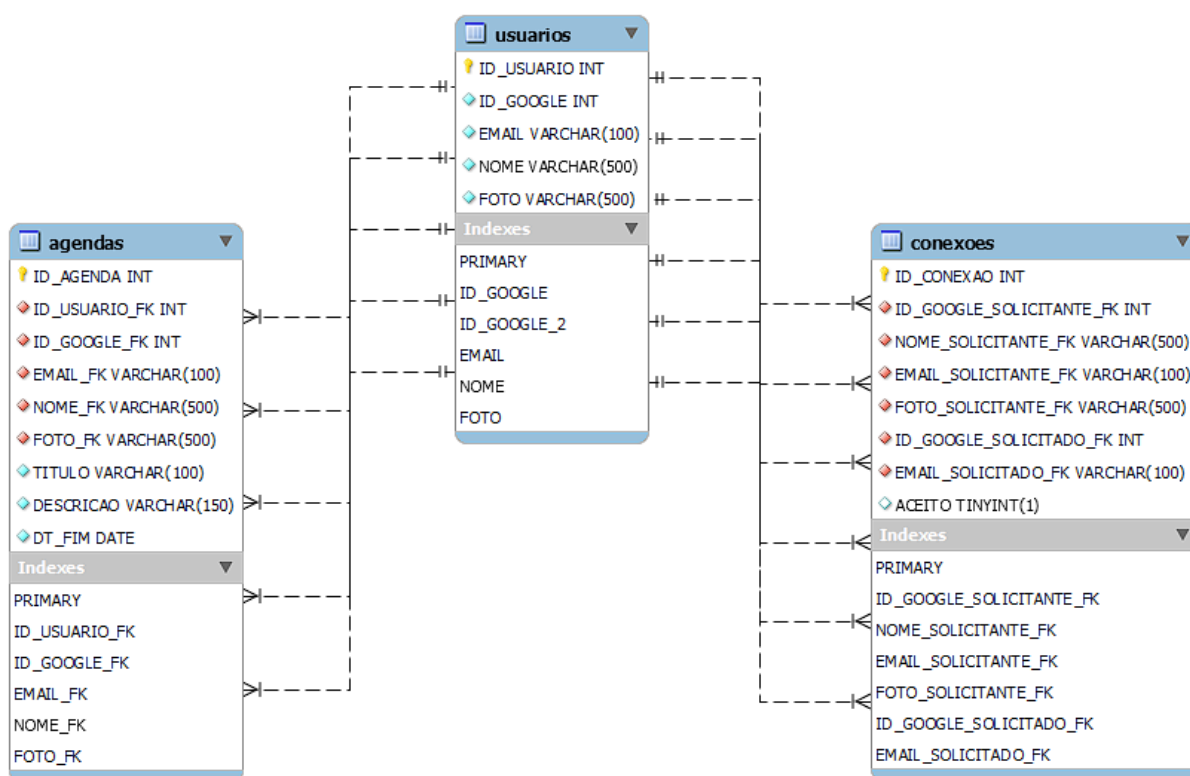


Figura 9 – Diagrama de Entidade Relacionamento do banco de dados.

4.5 Dicionário de Dados

Coluna	Tipo	Nulo	Comentários
ID_USUARIO	int	Não	Identificador da tabela USUARIOS.
ID_GOOGLE	int	Não	Id do usuário proveniente do Firebase.
EMAIL	varchar	Não	E-mail do usuário proveniente do Firebase.
NOME	varchar	Não	Nome do usuário proveniente do Firebase.

FOTO	varchar	Não	Foto do usuário proveniente do Firebase.
------	---------	-----	--

Tabela 15 – Dicionário de Dados da tabela USUARIOS.

Coluna	Tipo	Nulo	Comentários
ID_AGENDA	int	Não	Identificador da tabela AGENDAS.
ID_USUARIO_FK	int	Não	ID do usuário proveniente da tabela USUARIOS.
ID_GOOGLE_FK	int	Não	ID Google do usuário proveniente da tabela USUARIOS.
EMAIL_FK	varchar	Não	E-mail do usuário proveniente da tabela USUARIOS.
NOME_FK	varchar	Não	Nome do usuário proveniente da tabela USUARIOS.
FOTO_FK	varchar	Não	Foto do usuário proveniente da tabela USUARIOS.
TITULO	varchar	Não	Título da agenda.
DESCRICAÇÃO	varchar	Não	Breve descrição da agenda.
DT_FIM	date	Não	Data final da agenda, para controle do usuário.

Tabela 16 – Dicionário de Dados da tabela AGENDAS.

Coluna	Tipo	Nulo	Comentários
ID_CONEXAO	int	Não	Identificador da tabela CONEXOES.
ID_GOOGLE_SOLICITANTE_FK	int	Não	ID Google do usuário que solicita a conexão com outro usuário, proveniente da tabela USUARIOS.
NOME_SOLICITANTE_FK	varchar	Não	Nome do usuário que solicita a conexão com outro usuário, proveniente da tabela USUARIOS.
EMAIL_SOLICITANTE_FK	varchar	Não	E-mail do usuário que solicita a conexão com outro usuário, proveniente da tabela USUARIOS.
FOTO_SOLICITANTE_FK	varchar	Não	Foto do usuário que solicita a conexão com outro usuário, proveniente da tabela USUARIOS.
ID_GOOGLE_SOLICITADO_FK	varchar	Não	ID Google do usuário que receberá a solicitação de conexão, proveniente da tabela USUARIOS.
EMAIL_SOLICITADO_FK	varchar	Não	E-mail do usuário que receberá a solicitação de conexão, proveniente da tabela USUARIOS.

ACEITO	tinyint	Sim	Estado em que a conexão se encontra: NULL (quando está em aberto), TRUE (quando a conexão foi aceita) ou FALSE (quando a conexão foi recusada).
--------	---------	-----	---

Tabela 17 – Dicionário de Dados da tabela CONEXOES.

5 NOTAS DE ATUALIZAÇÕES FUTURAS

Sabemos que num ambiente real seria necessário uma série de atualizações para que nossa aplicação fosse vista com olhos mais positivos, seja na sua engenharia ou na visão o usuário, focamos aqui num MVP (Minimum Viable Product) para que pudéssemos lançá-lo no mercado, desta maneira, levantamos alguns pontos que seriam de possível visualização numa versão 1.1 do aplicativo:

1. Tokenização para requisições: Nossa aplicação não tem nenhum passo de autenticação que evite que Hackers ou softwares maliciosos acessem os Endpoints da API e insiram, deletem ou mude dados, ou até mesmo roubar dados. A Tokenização daria uma freada em ações maliciosas contra nossa aplicação.
2. Formulário de dados de cliente: Falamos anteriormente que a criação de formulários de dados é algo maçante e que queremos extingui-lo, todavia, em termos comerciais (algumas empresas compradoras de aplicativos evitam suas compras por não obter dados como endereço), ainda é necessário este tipo de formulário, nossa ideia seria resumi-lo ao mínimo, acrescentando somente nele alguns dados que não viriam no Google, como por exemplo o CEP, que retorna uma série de dados via alguma API de retorno de CEP.
3. Implementação de API de Consumo de CEP: Existem algumas APIs como o ViaCep (<https://viacep.com.br>) que nos retornam informações de localidades baseadas em um CEP fornecido. Todavia, este tipo de API é pago e teríamos que fornecer pagamento mensal para sua implementação, portanto, optamos por deixá-la fora da aplicação na versão do MVP.
4. Paginação da API: Imaginemos que uma pessoa tenha utilizado nossa aplicação por 3 anos e que por dia houvesse uma média de 3 agendas. Contaríamos um total de 3285 agendas para esta pessoa, agora imaginemos que ele tenha mais 9 conexões com os mesmos valores (hipoteticamente), estaríamos falando de 32850 agendas, requisitar estes dados todos de uma vez e renderizá-los em tela levaria cerca de 1,7 minutos (cálculo feito baseado na latência média da nossa aplicação). Então, provavelmente o usuário não iria aguardar 1,7 minutos até poder utilizar nossa aplicação. Paginando nossa aplicação para trazer de 50 em 50 resultados baseado na data mais próxima

de agendas poderíamos melhorar essa performance, e toda vez que o usuário chegasse ao fim da lista de agendas, mais 50 agendas seriam requisitadas.

5. Notificações de Push: Existem vários tipos de notificações que poderíamos pensar, todavia, algumas modalidades seriam bastante úteis pensando na frente do usuário. A primeira seria em relação a notificações de Pop-up informando quando um evento estivesse próximo. A segunda seria em relação a solicitações de pessoas, notificar quando alguém solicitou uma nova conexão ou aceitou a sua solicitação.
6. Outros métodos de Login: Além de Logins via Google, poderíamos também implementar conexões a aplicação via Facebook, GitHub, Slack, ou via login e senha padrão, neste último caso, o usuário prioritariamente deveria prescrever suas informações pessoais em um formulário.
7. Agendas Semanais: Sabemos que algumas agendas são diárias e acontecem todos os dias, por exemplo uma reunião com os amigos ou lavar a louça, as agendas semanais seriam uma mão na roda para que os usuários não tivessem que salvar todos os dias suas agendas.

6 CONSIDERAÇÕES FINAIS

Este trabalho de conclusão de curso foi realizado ao longo do último ano, e pôs em prova tudo aquilo que aprendemos durante o curso e em nossas empresas, o uso de bibliotecas e Framework Front-end foram abordadas principalmente na parte de criação da visualização das telas. A abordagem inicial seria feita via React padrão, produzindo uma aplicação Web, mas, sabemos que nossa aplicação não seria de grande uso dentro de um website, ligar um computador somente para acessar uma agenda é algo irreal de se pensar atualmente, portanto, optamos pela utilização do React Native que se baseia na linguagem de JavaScript e que é imensamente parecido com o React padrão, o que muda são algumas tags de estilo e maneiras como devemos estruturar o código de HTML, foi bastante interessante também a seleção do React Native, que gerou em mim particularmente (Otávio Barros) um desejo de trabalhar profissionalmente com a biblioteca futuramente, por seu fácil entendimento e a facilidade de produção de aplicações mobile sem precisar instalar programas notavelmente complexos e pesados como o Android Studio, e isso nos permitiu fazer um layout bastante agradável em pouco tempo, é notável que existem melhorias visuais que poderiam ser feitas, como loadings animados, e algumas regalias visuais, todavia, para uma aplicação produzida em pouco mais de três meses (fase de código) julgamos que atingimos o MVP do projeto. Na parte de Back-end optamos pela abordagem da criação de API utilizando ASP.Net Core 5 em conjunto com a linguagem de programação C#, que deixa nossa API mais robusta e coerente. O C# é uma linguagem fortemente typada, produzindo assim uma base de dados mais coerente e precisa, algo que é muito pedido pelas empresas que trabalham com Tecnologia da Informação atualmente.

REFERÊNCIAS

CANGUÇU, Raphael. **O QUE SÃO REQUISITOS FUNCIONAIS E REQUISITOS NÃO FUNCIONAIS.** Disponível em <https://codificar.com.br/requisitos-funcionais-nao-funcionais/#Requisitos_Funcionais>. Acesso em: 04/12/2021.

DEVMEDIA. **OS 4 PILARES DA PROGRAMAÇÃO ORIENTADA A OBJETOS.** Disponível em <<https://www.devmedia.com.br/os-4-pilares-da-programacao-orientada-a-objetos/9264>>. Acesso em: 02/05/2022.

LONGEN, Andrei. **O QUE É MYSQL? GUIA PARA INICIANTES.** Disponível em <<https://www.hostinger.com.br/tutoriais/o-que-e-mysql>>. Acesso em: 02/05/2022.

LUCIDCHART. **O QUE É UM DIAGRAMA DE CLASSE UML?** Disponível em <<https://www.lucidchart.com/pages/pt/o-que-e-diagrama-de-classe-uml>>. Acesso em: 05/05/2022.

QUALIBEST. **ENTENDA O QUE É PESQUISA QUALITATIVA E QUANTITATIVA.** Disponível em <<https://www.institutoqualibest.com/blog/dicas/entenda-o-que-e-pesquisa-qualitativa-e-quantitativa/>>. Acesso em: 20/11/2021.

UFCG. **DIAGRAMA DE CLASSES.** Disponível em <<http://www.dsc.ufcg.edu.br/~jacques/cursos/map/html/uml/diagramas/classes/classes2.htm>>. Acesso em: 05/05/2022.

APÊNDICE A - MANUAL DO USUÁRIO

A Agenda Connect conta com 6 interfaces, algumas delas contendo telas secundárias para inserção de dados ou atualização de tabelas. O aplicativo é fácil de usar e bastante intuitivo, para que pessoas de todos os nichos e níveis de aprendizado eletrônico possam utilizar com pouco esforço técnico.

Primeira tela: Nesta tela iremos fazer o Login em nossa aplicação, não há segredo algum, apenas o usuário se deparar com um botão remetendo ao conectar-se a aplicação, ao clicar irá ser redirecionado a uma tela onde irá fazer o login via Firebase do Google.

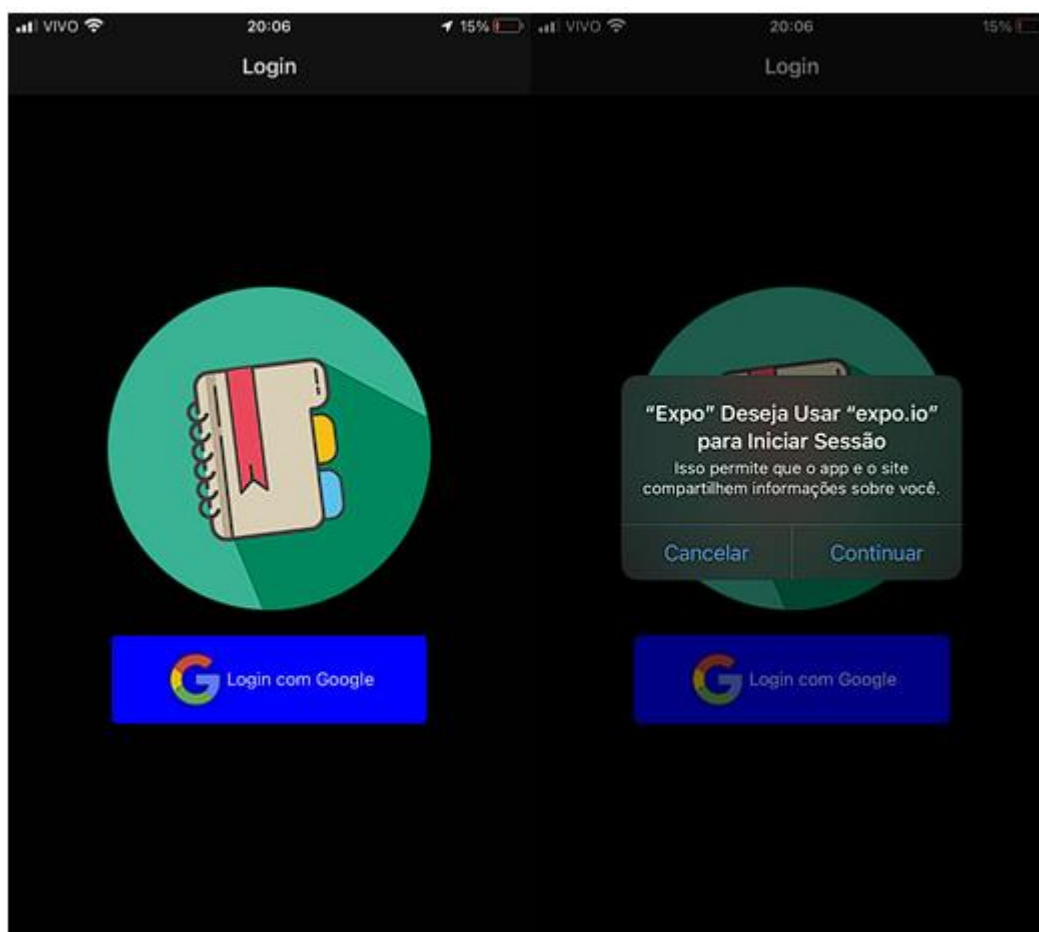


Figura 10 – Tela de login do sistema.



The app at **exp://192.168.0.237:19000/--/expo-auth-session** is asking you to sign into another service. Is this OK?



Figura 11 – Autorização para a sessão com o Expo.

Segunda tela: Após logado na aplicação o usuário será redirecionado para a tela de suas conexões, onde será possível ver uma lista de suas conexões ou uma tela dizendo que não há conexões ainda se for o caso, neste segundo será possível visualizar um botão em que ao ser clicado abre um modal para inserção de dados de um usuário para solicitar a conexão (existe também esse botão caso a página renderizada seja a de lista de conexões, porém este é menor, as cores são a mesma, o que pode ajudar o usuário a se localizar dentro da aplicação, existe também um ícone que remete a ação de adição de usuário neste segundo caso). Clicando em uma conexão é possível ver informações mais específicas do usuário assim como excluir a conexão. O modal desta tela é bastante simples e intuitivo também, basicamente será pedido o fornecimento de um “id de conexão” e “e-mail de conexão” que iremos citar posteriormente onde podemos encontrar essas informações.

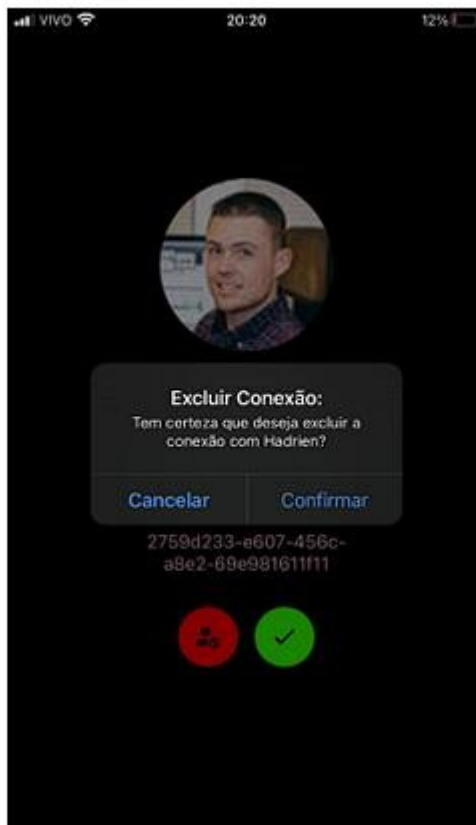
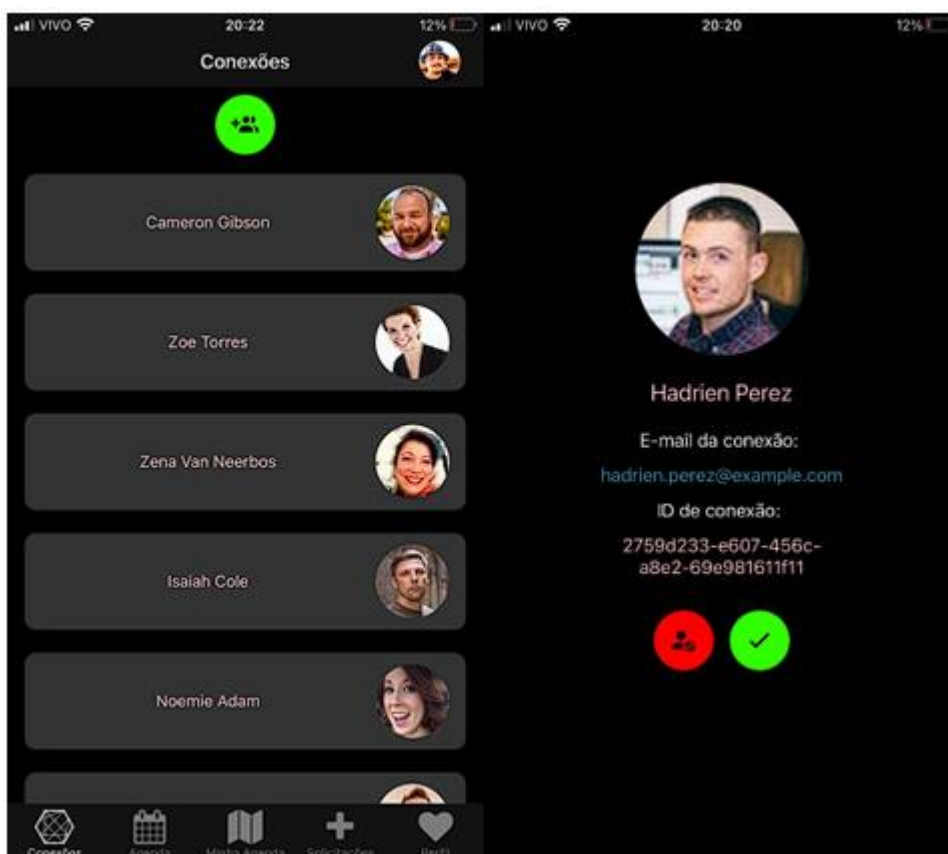


Figura 12 – Tela de conexões.

Terceira tela: Aqui é possível ver as agendas específicas do seu usuário, é possível também alterar ou excluir a agenda clicando em cada uma delas, ao abrir o modal que permite exclusão ou alteração é possível também ver informações mais específicas da agenda como data e o descritivo da agenda.



Figura 13 – Tela de agendas.

Quarta tela: Nesta página será possível ver uma lista de todas as agendas, tanto suas como das conexões, será disposto em um calendário onde é possível selecionar datas específicas e informações sobre os usuários donos da agenda. Clicando em cada uma dessas agendas é possível ver informações mais específicas sobre as agendas, mas não as atualizar ou excluí-las.

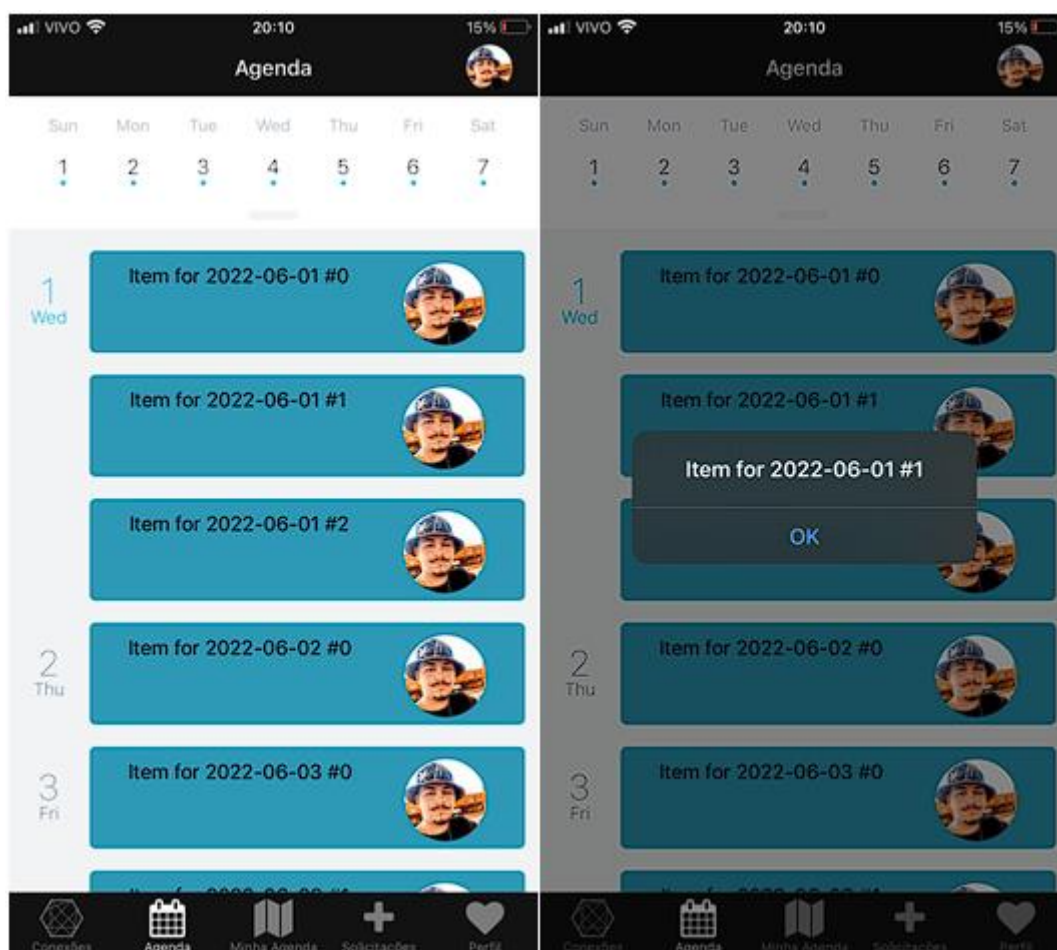


Figura 14 – Tela de calendário com todas as agendas.

Quinta tela: Aqui é possível ver as pessoas que solicitaram conexão com você, clicando em cada pessoa é possível aceitar ou recusar a conexão.

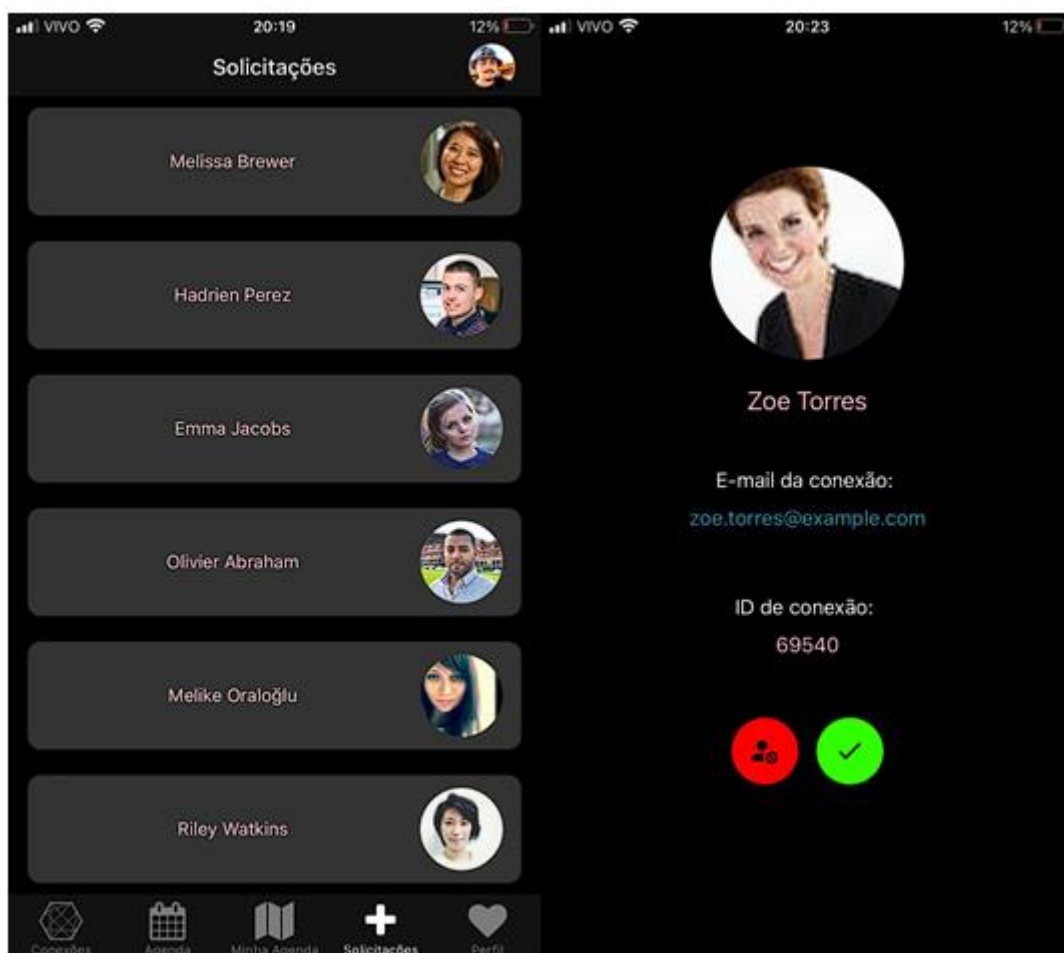


Figura 15 – Tela com solicitações de conexões.

Sexta tela: Aqui é possível ver suas informações pessoais, é possível visualizar seu “e-mail de conexão” e “id de conexão” nesta tela, assim como um descritivo/passos-a-passo de como se conectar a outra pessoa.

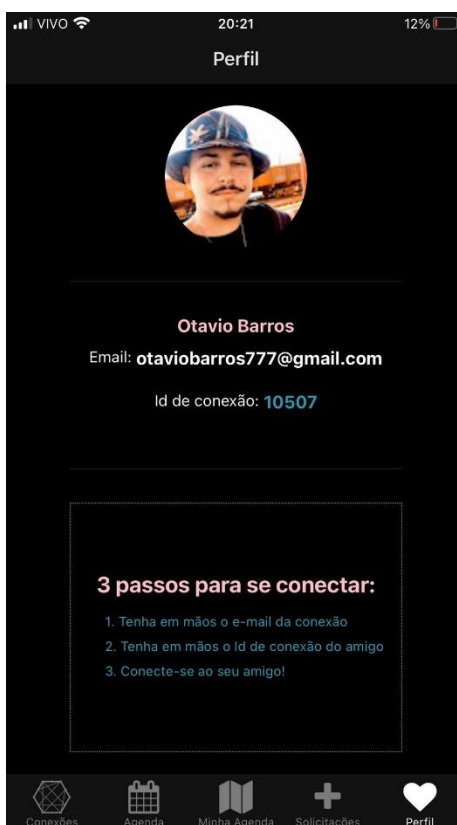


Figura 16 – Tela com informações do usuário.