Escola Politécnica da Universidade de São Paulo

PCS3100 - INTRODUÇÃO À ENGENHARIA DE COMPUTAÇÃO

Integrantes (Equipe 3)

Arthur Macedo de Alencar (14754219)

Enzo Koichi Jojima (14568285)

Vinícius Yshimine Terra (5306726)

Professor(es)

Edson Satoshi Gomi

Felipe Valencia de Almeida

Relatório de Conclusão de Trabalho do Trimestre

São Paulo

2023

Projeto Cronos

Idealizadores

Arthur Macedo de Alencar

Enzo Koichi Jojima

Vinícius Yshimine Terra

ÍNDICE DETALHADO

1. Apresentação

1.1. Temática

O projeto Cronos é uma aplicação de software (por enquanto) mobile voltada, primariamente, ao ambiente acadêmico, com utilidade especial (mas não restrita) à rotina universitária.

1.2. Problema a ser resolvido

A vida de universitário é frequentemente caótica: trabalhos a ser entregues, provas para estudar e diversas outras responsabilidades. Nesse âmbito, estudantes constantemente encontram dificuldades em coordenar seus muitos afazeres e, consequentemente, obtêm resultados pouco satisfatórios em seus percursos acadêmicos; problema, em grande parte, devido à simples desorganização da rotina de estudos.

1.3. Objetivo do projeto

Tendo em vista tais complicações, o objetivo do Projeto Cronos é oferecer uma luz no fim do túnel para esses acadêmicos. A plataforma tem o intuito de auxiliá-los a organizar suas diferentes tarefas fornecendo uma gama de ferramentas eficientes, como o registro de grade horária, a atribuição de tarefas/lembretes, um cronômetro, a utilização do timer pomodoro e a integração à nuvem.

1.4. Motivação para escolha de tema

Os desenvolvedores por trás do projeto Cronos tem grande familiaridade com a problemática introduzida e, por lhes ser uma realidade frequente, decidiram utilizar e incrementar seus conhecimentos de programação para criar uma solução para esse empecilho em suas rotinas. Em suma, trata-se de um trabalho feito por estudantes e destinado a estudantes.

1.5. Documentação

Este documento será utilizado para registrar a progressão do grupo ao longo do desenvolvimento do trabalho, desempenhando função de documento geral do projeto, bem como de relatório. Em vista da falta de experiência da equipe de desenvolvimento com a estrutura desse tipo de documento, a referência para sua produção será o modelo criado e disponibilizado na internet, em 2005, pela Profª Ana Paula Gonçalves Serra, da Universidade São Judas Tadeu, sob o título "Documentação de um Produto de Software".

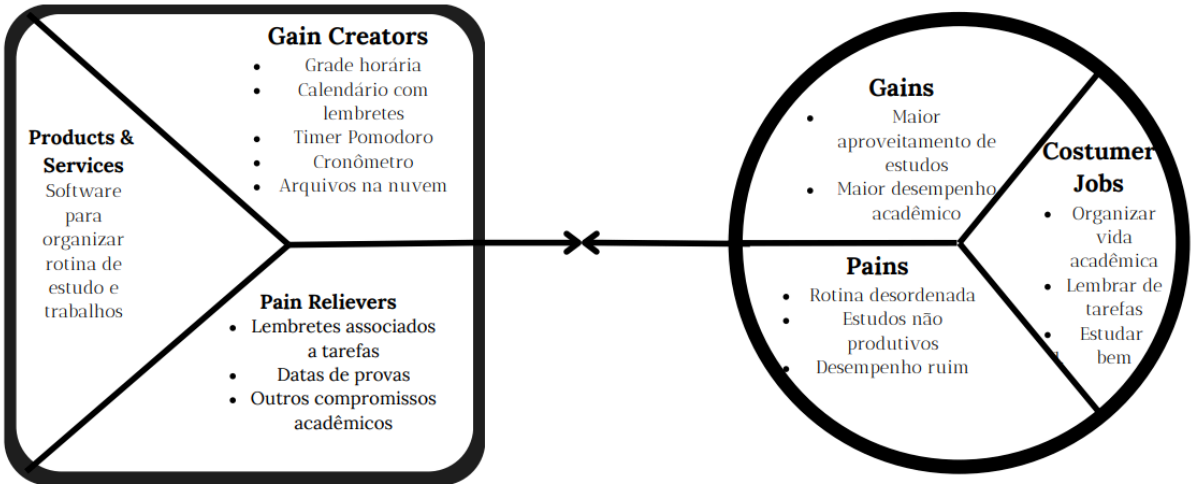
2. Descrição Geral do Sistema

2.1. Perfil dos usuários-alvo

As pessoas às quais o projeto Cronos se destina são estudantes, preferencialmente universitários, que possuem dificuldades de planejamento e execução de rotina de estudos e, consequentemente, têm desempenho acadêmico prejudicado por tal conjuntura.

2.2. Canvas de Osterwalder e soluções para problema

Com base na análise do perfil dos usuários, o seguinte diagrama foi desenvolvido pensando nas soluções para os problemas dos usuários:



Canvas de Osterwalder

Como listado acima, as soluções para lidar com o problema maior envolvem o controle maior da rotina dos prejudicados, o qual se dá por intermédio de diversas ferramentas para os auxiliar em suas tarefas acadêmicas.

Dentre as ferramentas selecionadas, estão 1) Grade horária, 2) Calendário editável, 3) Timer Pomodoro e 4) Cronômetro. As duas primeiras soluções oferecem uma perspectiva maior acerca da rotina do usuário e são modificáveis para acomodar as suas preferências, enquanto que as duas últimas possuem escopo diário, tendo aplicação em

atividades nas quais o controle do tempo é necessário. Com isso, a soma dessas funcionalidades em um único programa garante um produto que permite melhor planejamento e aproveitamento de estudos.

2.3. Impacto do projeto

Com base no tópico anterior, o grupo acredita que o conjunto das ideias apresentadas em um software mobile fará diferença significativa na vida acadêmica dos usuários ao proporcionar-lhes ferramentas de organização. A integração do método pomodoro e o maior controle da rotina de estudo garantirá maior produtividade e, consequentemente, desempenho aprimorado.

2.4. Desenvolvedores do projeto

A equipe por trás do projeto é composta por três estudantes de engenharia da computação na Escola Politécnica da Universidade de São Paulo (POLI-USP), os quais buscam, por meio deste trabalho, aprimorar seus conhecimentos sobre desenvolvimento mobile, e programação no geral, criando uma solução para um problema recorrente no ambiente em que estão inseridos.

3. Requisitos

3.1. Requisitos funcionais

A. Banco de dados na nuvem: a aplicação terá integração com o Google Firebase para armazenar as informações das contas dos usuários, permitindo que sejam salvas no app ao o acessarem novamente.
B. Grade horária: essa ferramenta serve para o usuário ter uma perspectiva geral do seu dia-a-dia acadêmico, podendo alterá-la ao seu critério.
C. Calendário editável: será disposto em uma página na qual haverá a opção de atribuir tarefas, lembretes, prazos, datas importantes etc. Para eventos, o usuário pode delimitar sua duração, e o aplicativo apresentá-lo na Home Page.
D. Telas e sistemas de cadastro e login: aproveitando-se da conexão com Firebase, essas funções garantem que somente o usuário possa ter acesso a sua conta Cronos.
E. Timer Pomodoro: baseado na famosa "Técnica Pomodoro", trata-se de duas opções de timer, um de 25 minutos para estudo focado, outro de 5 para intervalo de descanso.
F. Cronômetro: é uma ferramenta mais geral para conveniência; pode ser usada para marcar tempo durante simulados ou resolução de exercícios. As possibilidades são vastas.

3.2. Requisitos não funcionais

A. Portabilidade: aplicação mobile, inicialmente, para sistemas Android.
B. Interface intuitiva: o frontend da aplicação é pensado na facilidade de uso para satisfazer as necessidades dos mais diversos usuários.
C. Relevância: o framework acessível e completo do Flutter confere à aplicação a possibilidade de receber futuras atualizações que adicionem novas funcionalidades.

D. Navegação otimizada: o usuário possui a liberdade de locomoção para qualquer página a partir de qualquer página do app.

3.3. Prototipação

3.3.1. Descrição das telas

Login

Funções: autenticar o e-mail e a senha do usuário para, caso corretos, levá-lo à Home. Caso seja a primeira vez em que o app é inicializado, há a possibilidade de ir para a tela de cadastro para a criação de uma nova conta.

Direciona a: Cadastro, Home.

Cadastro

Funções: possibilitar a criação de uma nova conta para o usuário (arquivada no Firebase) e, após isso, retornar à tela de Login.

Direciona a: Login.

Home

Funções: página de "boas-vindas" do app. Caso o usuário esteja conectado, sempre será a tela apresentada após a inicialização.

Direciona a: telas conectadas à barra de navegação.

Cronômetro

Funções: apresentar um timer que pode ser inicializado, parado ou reiniciado.

Direciona a: telas conectadas à barra de navegação.

Grade Horária

Funções: permitir a adição de matérias de modo a criar um cronograma acadêmico para o usuário.

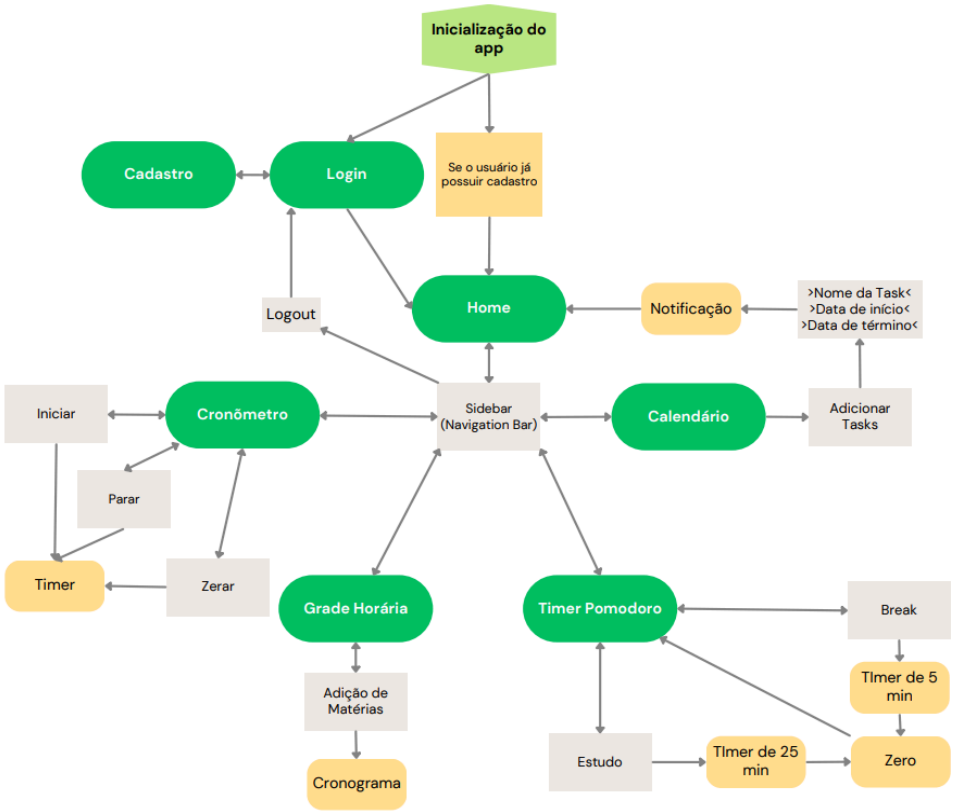
Direciona a: telas conectadas à barra de navegação.

Timer Pomodoro

Funções: apresentar duas opções de timer, com pausa, para a escolha do usuário, “estudo” e “break”.

Direciona a: telas conectadas à barra de navegação.

3.3.2. Diagrama de navegação



4. Implementação

4.1. Framework e ferramentas de trabalho

O desenvolvimento completo da aplicação será baseado nas ferramentas do framework Flutter, que é ideal para a criação de aplicações mobile - neste caso, a programação será feita utilizando a linguagem Dart. Para editar o código, é feito uso do editor Visual Studio Code. Além disso, para poder visualizar a execução do aplicativo é necessária a instalação do programa Android Studio e, posteriormente, a criação de um emulador de sistema Android (recomenda-se o Pixel XL API 30). Enfim, para testar a execução do programa na prática, recomenda-se a posse de um dispositivo para inicializar a aplicação protótipo (o grupo optou por um celular modelo Motorola G1).

Com as ferramentas apresentadas, e seu domínio, o desenvolvimento da aplicação pode começar.

4.2. Cronograma

Para aumentar a produtividade e registrar uma linha temporal do desenvolvimento, o grupo criou um cronograma simples repartindo as etapas de confecção do app no tempo de semanas conforme ilustra a imagem abaixo.

Semana 1 21/5 - 28/5	Semana 2 28/5 - 03/06	Semana 3 04/06 - 10/06	Semana 4 11/06 - 17/06	Semana 5 18/06 - 24/06	Semana 6 25/06 - 01/07
Pesquisa sobre utilização do flutter	Início do desenvolvimento do app	Implementação da conexeção com o Firebase	Tela Home funcional	Tela Grade Horaria funcional	Testes finais
Pesquisa sobre uso do banco de dados na núvem	Familiarização com o framework Flutter	Tela de Login/Cadastro	Tela Calendario funcional	Tela Timer funcional	Vídeo explicando projeto
Pesquisa sobre a linguagem Dart	Familiarização com a linguagem Dart	Sistema de Login/Cadastro		Testes iniciais	Finalização do relatório

4.3. Programação

- Pacotes necessários

```
cloud_firestore: ^4.7.1
cupertino_icons: ^1.0.2
firebase_auth: ^4.6.1
firebase_core: ^2.13.0
firebase_storage: ^11.2.1
flutter:
  sdk: flutter
intl: ^0.18.1
```

```
material_design_icons_flutter: ^7.0.7296
table_calendar: ^3.0.9
timezone: ^0.9.2
```

Os pacotes acima são precisos, uma vez que permitem a programação das funções implementadas no app.

- Integração ao firebase - login e cadastro

```
//signup the user function
Future<String> signUpUser({
  required String email,
  required String username,
  required String password,
}) async {
  String res = 'Some error ocurred';
  try {
    if (email.isNotEmpty || username.isNotEmpty || password.isNotEmpty) {

      //register the user
      UserCredential cred = await _auth.createUserWithEmailAndPassword(
          email: email, password: password);

      if (kDebugMode) {
        print(cred.user!.uid);
      }

      //adding user to our database
      await _firestore.collection('users').doc(cred.user!.uid).set({
        'email': email,
        'uid': cred.user!.uid,
        'username': username,
        'password': password,
      });
      res = 'User registered successfully';
      if(res == 'User registered successfully'){
        User? user = _auth.currentUser;
        if(user != null) {
          await user.updateDisplayName(username);
        }
      }
    }
  } on FirebaseAuthException catch (e) {
    if (e.code == 'weak-password') {
      res = 'The password should have at least 6 characters.';
    } else if (e.code == 'email-already-in-use') {
      res = 'The account already exists for that email.';
    } else if (e.code == 'invalid-email') {
      res = 'The email address is not valid.';
    }
  } catch (e) {
    res = e.toString();
  }

  return res;
}
```

```
//loggin the user function
Future<String> logginUser({
  required String email,
  required String password,
}) async {
  String res = 'Some error ocurred';
  try {
    if (email.isNotEmpty || password.isNotEmpty) {
      //login the user
      UserCredential cred = await _auth.signInWithEmailAndPassword(
          email: email, password: password);

      if (kDebugMode) {
        print(cred.user!.uid);
      }

      res = 'User logged in successfully';
    }else{
      res = 'Please fill all the fields';
    }
  } on FirebaseAuthException catch (e) {
    if (e.code == 'user-not-found') {
      res = 'No user found for that email.';
    } else if (e.code == 'wrong-password') {
      res = 'Wrong password provided for that user.';
    } else if (e.code == 'invalid-email') {
      res = 'The email address is not valid.';
    }
  } catch (e) {
    res = e.toString();
  }

  return res;
}
```

A função de signUpUser cria um documento no firebase contendo email, senha, nome de usuário e uid (user ID). O uid e email são únicos para cada usuário. Além disso, ela também cria um usuário no Firebase Authentication com email e senha.

Já a função loginUser irá user essa autenticação com email e senha para autenticar o usuário dentro do app.

- Routing

```
void _logout(BuildContext context) async {
  await _firebaseAuth.signOut();
  changeScreen(context, LoginScreen());
}

void _navigateToCronometro(BuildContext context){
  Navigator.push(
    context,
    MaterialPageRoute(builder: (context) => StopwatchPage()),
  );
}

void _navigateToCalendar(BuildContext context) {
  Navigator.push(
    context,
    MaterialPageRoute(builder: (context) => CalendarPage()),
  );
}

void _navigateToHome(BuildContext context) {
  Navigator.push(
    context,
    MaterialPageRoute(builder: (context) => HomeScreen()),
  );
}


void _navigateToGrade(BuildContext context) {
  Navigator.push(
    context,
    MaterialPageRoute(builder: (context) => GradeHorariaScreen()),
  );
}

void _navigateToPomodoro(BuildContext context) {
  Navigator.push(
    context,
    MaterialPageRoute(builder: (context) => PomodoroPage()),
  );
}
```

Essas funções são chamadas pelos botões na sidebar, que cumpre o papel de routing global. Elas contam com a função Navigator.push(), nativa do próprio Flutter. Já a função logout usa a função signOut do package firebase_auth.

- Funcionalidades

Tasks e Grade Horária

A implementação das tasks e aulas foi bem parecida. Ambas são coleções dentro do firebase, nas quais cada task/aula é um documento diferente.

Para implementarmos, devemos criar funções que criam coleções e documentos dentro do firebase, além de conseguirem editar

o conteúdo desses documentos, excluir tais documentos e extrair os dados desses documentos, a fim de organizar as tasks e aulas na tela de maneira clara. Lembre-se de que as coleções devem ser criadas dentro do documento de cada usuário. Dessa forma, cada usuário terá sua própria lista de tasks e aulas.

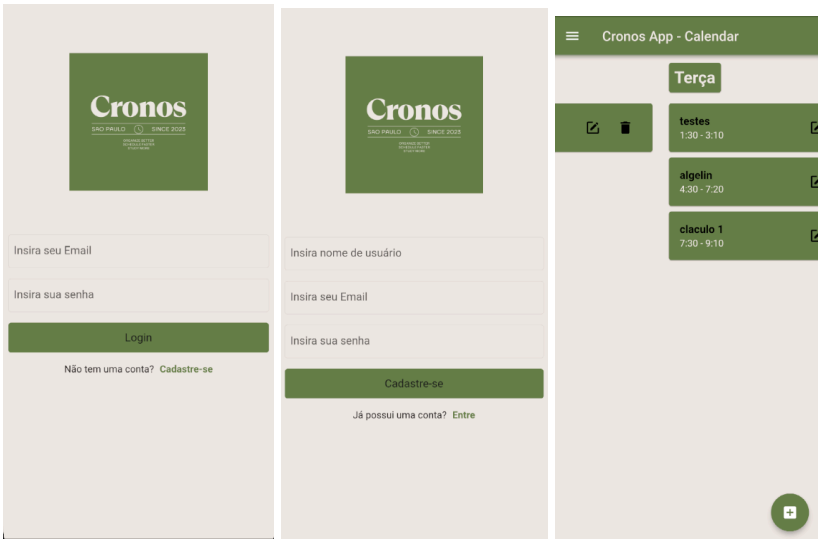
Pomodoro e Cronometro

Para o código do cronômetro, criamos uma funcionalidade no Flutter com dois botões, "Start/Stop" e "Reset", e um widget de texto para exibir o tempo decorrido. A funcionalidade do cronômetro é obtida usando uma combinação das classes setState e Timer no Flutter.

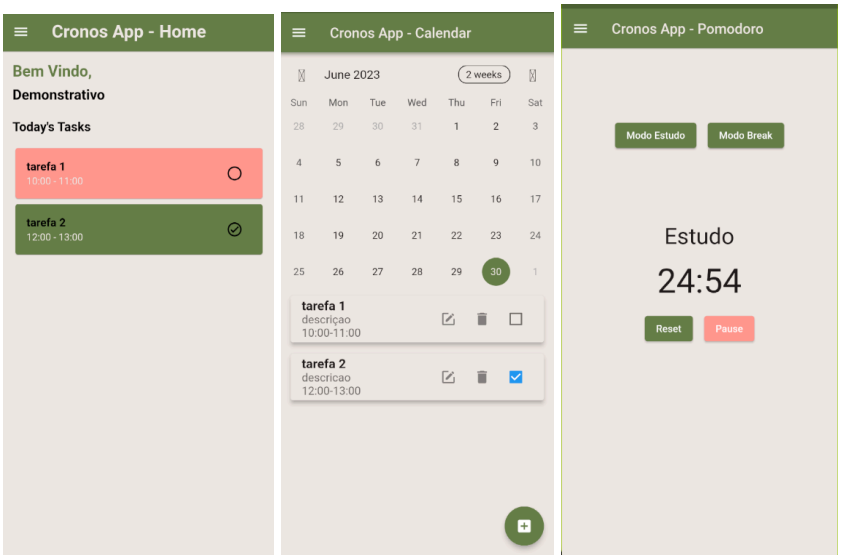
A técnica Pomodoro é um método de gerenciamento de tempo que usa um cronômetro para dividir o trabalho em intervalos, tradicionalmente de 25 minutos, separados por intervalos curtos.

No código da funcionalidade Pomodoro, temos um widget PomodoroPage que gerencia o estado do timer Pomodoro. Ele usa um Timer para atualizar o tempo restante a cada segundo. O cronômetro alterna entre durações de trabalho e pausa com base no sinalizador _isWorking. O usuário pode iniciar, pausar, redefinir o cronômetro e a interface do usuário reflete o estado atual.

```
void resetTimer() {
  setState(() {
    isPaused = false;
    isRunning = false;
    milliseconds = 0;
  });
}

void toggleTimer() {
  setState(() {
    isRunning = !isRunning;
  });
  if (isRunning) {
    _animationController.forward(from: 0.0);
    isPaused = false;
  } else {
    isPaused = true;
    _animationController.stop();
  }
}

String formatMilliseconds(int milliseconds) {
  int seconds = (milliseconds ~/ 1000) % 60;
  int minutes = (milliseconds ~/ (1000 * 60)) % 60;
  int hours = (milliseconds ~/ (1000 * 60 * 60)) % 24;

  String secondsStr = seconds.toString().padLeft(2, '0');
  String minutesStr = minutes.toString().padLeft(2, '0');
  String hoursStr = hours.toString().padLeft(2, '0');

  return "$hoursStr:$minutesStr:$secondsStr";
}
```



Acima, telas de login e registro e grade horária, respectivamente.



Acima, telas de home, calendário e pomodoro, respectivamente.

- Design

A aplicação busca transmitir modernidade e simplicidade em sua aparência, de modo a atingir um ideal de eficiência. Para o ícone do app nos celulares, foi criada a imagem abaixo:

A imagem do relógio, combinada ao ‘C’ de “Cronos”, indica a ideia de produtividade e/ou tarefa. A escolha de cor faz referência à paleta padrão escolhida pelo grupo para representar o projeto, composta pelos tons #667f48 e #efe8e3, os quais serão aplicados ao resto da interface do app.

O design do app foi montado a partir da framework do flutter, que possui ferramentas de desenvolvimento front-end baseadas na programação orientada a objetos, centrada em seus widgets, que cumprem o papel de peças num grande quebra-cabeças.

5. Testagem

5.1. Plano de teste

Os testes têm o fito de verificar a execução do programa desenvolvido, bem como atestar a eficácia das funcionalidades da aplicação. Para tal, o seguinte planejamento foi realizado:

1. **Instalação da aplicação no dispositivo**

   Descrição: o usuário deverá, por meio de uma transmissão USB, importar a aplicação para o seu celular, localizar o ícone da aplicação e inicializá-la.

   Resultado esperado: a importação e inicialização do app serão bem sucedidos e o usuário chegará na tela de cadastro.

2. **Cadastro, logout e login**

   Descrição: após a inicialização do app, o usuário deve, primeiramente, realizar seu cadastro e, posteriormente, seu login, depois do qual, será direcionado à home page. Ademais, ele deve testar a opção de logout e, com isso, a de login novamente com o intuito de testar a integração com o google firebase.

   Resultado esperado: o usuário será capaz de criar sua conta, realizar login nela e ser direcionado à homepage; após isso, ao selecionar a opção de "logout", localizada no canto inferior da barra de navegação, será novamente levado à tela de login e o fará, parando, enfim, na home page mais uma vez.

3. **Navegação**

   Descrição: o usuário precisa navegar pela aplicação, acessando todas as páginas disponíveis e verificando se é direcionado corretamente a tais e se realiza o caminho inverso com êxito.

Resultado esperado: na home page, ao selecionar a barra de navegação, o usuário terá a liberdade para transitar, com sucesso, entre as páginas da aplicação e, eventualmente, retornar à home page.

4. **Funcionalidades**

   Descrição

   - Grade Horária: o usuário deve criar um cronograma a partir da distribuição de tarefas/matérias ao longo da semana.
   - Calendário: ao acessar essa página, o usuário deve ter, em sua tela, um calendário com a marcação do dia em que está usando o aplicativo. Ao selecionar qualquer data, o usuário deve ser capaz de adicionar tarefas e configurar sua duração de tempo. Além disso, ao retornar à Home, a tarefa adicionada precisa ser mostrada, indicando se ela está atrasada ou dentro do prazo escolhido.
   - Timer Pomodoro: devem estar disponíveis duas opções de timer para o usuário escolher. Ao selecionar “Estudo”, um timer de 25 minutos deve ser inicializado; ao optar pela outra ação, “Break”, um timer de 5 minutos irá começar. Essa oportunidade deve ser aproveitada para averiguar a resposta dos timers ao botão de parada.
   - Cronômetro: essa ferramenta deve responde de imediato às ações do usuário; no caso, deve-se verificar se o cronômetro está marcando a variação de tempo na velocidade certa e se as opções de parar e reiniciar funcionam.

     Resultado esperado: todas as funções implementadas serão executadas com sucesso: o usuário será capaz de personalizar seu calendário com seus lembretes e montar sua grade horária; além disso, os timers farão a marcação de tempo corretamente e responderão de imediato às ações do usuário.

5.2. Execução de testes

Devido aos limites relacionados à disponibilidade de dispositivos teste, foi realizado um teste da aplicação no seguinte dispositivo: Motorola G1.

O resultado do teste está disposto abaixo:

Teste n°1

Data: 28/06/2023

Dispositivo: Motorola G1, Ver: 5.1.

Resultados: O aplicativo foi executado com sucesso. Todas as funções foram realizadas com êxito, conforme os resultados esperados. Contudo, em termos de visualização, a grade horária estava demasiadamente grande para a tela do dispositivo.

Observações: Apesar de terem sido importados durante a programação, os ícones não foram carregados para a aplicação. A tela da grade horária ainda precisa passar por ajustes, pois não se adequou ao tamanho da tela do dispositivo.

Teste n°2

Data: 30/06/2023

Dispositivo: Motorola G1, Ver: 5.1.

Resultados: A execução do aplicativo teve sucesso novamente. De fato, todas as funcionalidades obtiveram êxito, tal qual esperava-se. O ícone do aplicativo, que antes era ausente, agora aparece com êxito na tela de seleção de apps do smartphone.

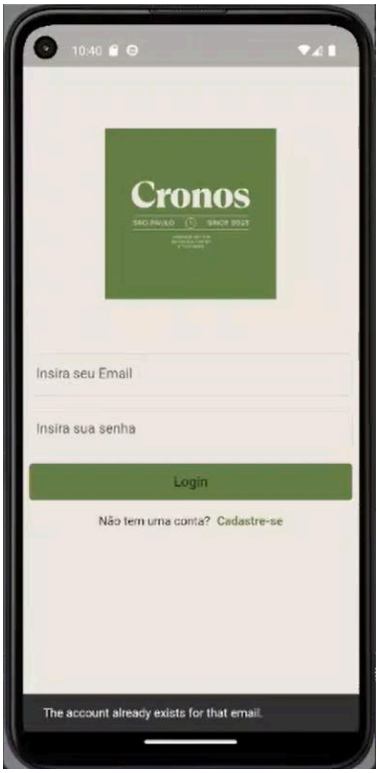
Observações: A grade horária, anteriormente, estava experienciando erros, mas agora funciona normalmente; além disso, os ícones estão sendo carregados devidamente.

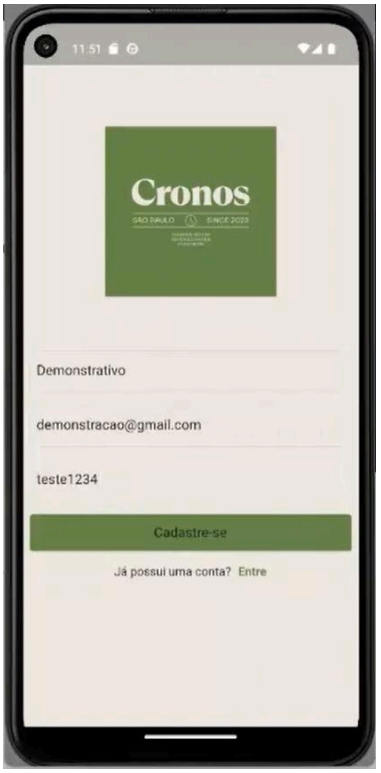
6. Manual do Usuário

Em vista dos testes bem sucedidos, e após os ajustes finais, a aplicação se encontra em estado funcional e pronta para utilização. Sendo assim, os desenvolvedores apresentam o seguinte guia geral para auxiliar durante o uso do app Cronos.

6.1. Cadastro e login do usuário

Ao inicializar Cronos, caso o usuário não esteja conectado a sua conta, ele irá se deparar com a tela de Login. Nela há espaços para a inserção dos dados da conta a serem verificados pelo app (e-mail e senha). Se for a primeira vez do usuário utilizando a aplicação, ele deve selecionar a opção “Cadastre-se”, a partir da qual será direcionado à tela de cadastro, em que ele deve fornecer um e-mail e criar uma senha para sua nova conta Cronos. Tais informações são armazenadas no Firebase para que essa pessoa possa retornar ao login e, após fazê-lo, acessar a Home.





6.2. Home e barra de navegação

Caso o usuário tenha atribuído tarefas ou eventos no calendário anteriormente, elas deverão aparecer na Home: verde, se ainda estiverem dentro do prazo, ou vermelha, se a data limite foi ultrapassada.

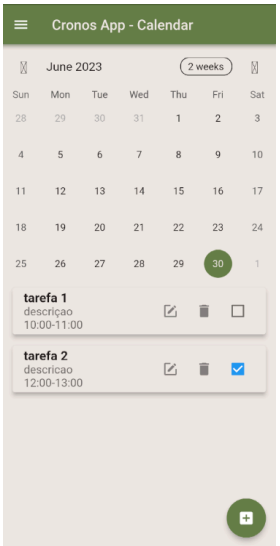
No canto superior esquerdo, o ícone das “três barrinhas” indica o acesso à barra de navegação, que corresponde ao veículo de navegação do usuário pelo app. É importante notar que sua última opção contém o botão “Logout”, o qual, ao ser ativado, retorna o usuário à tela de Login.

Essa barra de navegação é constante para todas as páginas do app, permitindo circulação livre e fácil por ele.

6.3. Grade horária e calendário

Acessando a grade horária, o usuário pode adicionar tarefas ou matérias ao longo dos dias da semana.

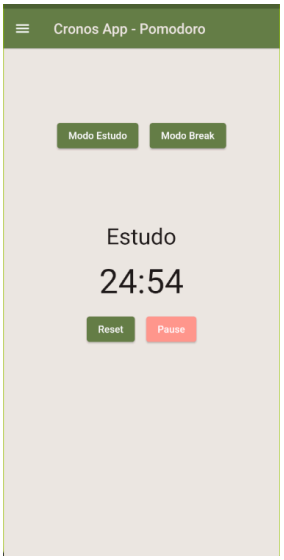
Já na tela do calendário, há a datação padrão. Selecionando uma data qualquer, é possível adicionar um evento nela de forma a determinar seu nome, descrição e duração. Tal tarefa passará a ser mostrada na Home toda vez que o app for inicializado. Existe a opção de removê-la tanto nesta tela quanto na de calendário.



6.4. Cronômetro e timer Pomodoro

Na página do cronômetro, o usuário possui a liberdade de inicializá-lo, pará-lo ou reiniciá-lo. É uma ferramenta simples, porém com bastante aplicabilidade.

Por fim, o timer Pomodoro, baseado na metodologia de aprendizado homônima, disponibiliza a opção de estudo focado e ininterrupto, representado pelo botão “Estudo”, além de um intervalo de repouso. Quando a primeira escolha é selecionada, um timer decrescente de 25 minutos é inicializado, já na segunda, um de 5 minutos; ambos podem ser parados ou reiniciados a qualquer instante.



7. Conclusão

7.1. Resultado do projeto

Com a finalização do trabalho, cabe um breve relatório da experiência dos desenvolvedores durante a criação do app Cronos:.

A escolha de uma temática não foi questão de conflito, pois é um tópico presente no cotidiano da equipe. No entanto, houve diversas dúvidas quanto à implantação de IoT externo. No começo, tinha-se a pretensão de criar uma aplicação controlada por uma engine de reconhecimento de voz, contudo, o grupo não conseguiu pôr a ideia à frente e decidiu focar somente no projeto de software.

O primeiro contato com o Flutter foi difícil, o grupo encarou o desafio de dominar suas ferramentas por conta própria. Com o auxílio da internet, os desenvolvedores conseguiram estudar o suficiente do framework e da linguagem para dar início à fase de programação.

A implementação e fase de teste da aplicação foram bem sucedidas, apesar de algumas complicações iniciais. Mesmo que várias ideias tenham sido descartadas ao longo do desenvolvimento do app, o resultado final provou ser bastante satisfatório para a questão que busca enfrentar.

É certo que o trabalho produzido incentivou os desenvolvedores a se afastarem de sua zona de conforto e aprenderem uma nova linguagem, assim como uma nova metodologia de desenvolvimento de software. Ao final, a equipe tem orgulho dos resultados obtidos e espera usar a experiência adquirida em futuros projetos.

Portanto, o grupo logrou em desenvolver o app Cronos, que combate o problema da desorganização da rotina de estudos originalmente apresentado no tópico 1.1 deste documento, tornando o projeto, assim, um sucesso.

7.2. Expectativas futuras

Após esta primeira etapa de desenvolvimento, em face do potencial apresentado pela aplicação, o grupo acredita que, sob condições mais favoráveis de tempo, é possível incrementá-la. Abaixo são listadas algumas ideias dos desenvolvedores que não apareceram na versão final do app:

A. Porte para sistema iOS
    - Por enquanto, o app Cronos tem exclusividade para sistemas Android, contudo, existe a possibilidade para portá-lo para o sistema IOS, o que aumentaria seu alcance de ação e seu impacto.

B. Novas funcionalidades
    - O app Cronos foi pensado para continuar relevante a seus usuários por meio de atualizações. O framework do Flutter possibilita a efetividade de tal ação. Assim, no futuro, o app pode receber novas funcionalidades que potencializem sua ação.

C. Hardware externo
    - Devido a questões fora do campo do desenvolvimento do app, não foi possível a implantação de um hardware externo; contudo, os desenvolvedores tiveram ideias que, caso realizadas, ampliariam a ação do app Cronos (a exemplo de uma função que determinava a adequabilidade das condições de uma sala para estudo conforme parâmetros disponibilizados previamente pelo usuário).

8. Bibliografia

Gonçalves, Ana Paula. “Documentação de um Produto de Software.” *Prof. Walteno Martins Parreira Júnior*, Disponível em: <http://www.waltenomartins.com.br/esof2_projeto_documento_de_sw.pdf>. Accessed 30 June 2023.

AHMED, Y. Flutter authentication: Implementing user signup and Login. Disponível em: <https://www.loginradius.com/blog/engineering/guest-post/authenticating-flutter-apps/>. Acesso em: 30 jun. 2023.

Firebase_auth, 2023. Disponível em: <https://pub.dev/packages/firebase_auth>. Acesso em: 30 jun. 2023.

Layouts in flutter. Disponível em: <https://docs.flutter.dev/ui/layout>. Acesso em: 30 jun. 2023.

RANAWAT, R. Instagram Clone. Disponível em: <https://github.com/RivaanRanawat/instagram-flutter-clone>. Acesso em: 30 jun. 2023.

table_calendar | Flutter Package. Disponível em: <https://pub.dev/packages/table_calendar/>. Acesso em: 30 jun. 2023.

How can I create a custom calendar in Flutter? Disponível em: <https://stackoverflow.com/questions/61755268/how-can-i-create-a-custom-calendar-in-flutter>. Acesso em: 30 jun. 2023.