Anima Educação – Escola de TI e Computação

Bacharelado Em Ciência Da Computação

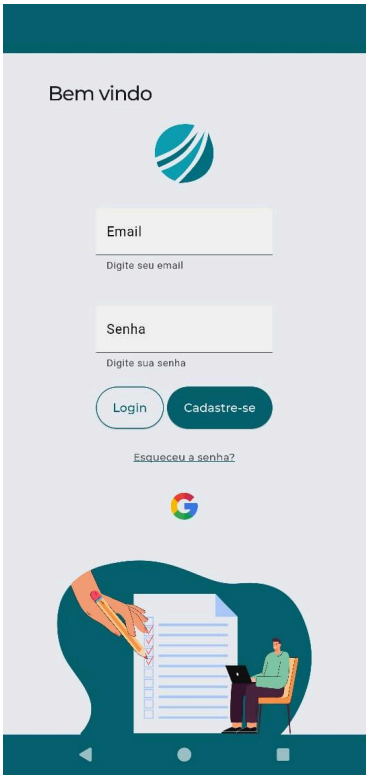
CST em Análise e Desenvolvimento de Sistemas

Caique da Silva Miranda Santos - 12723216257
Hercules Matheus Lima Silva - 12823214720

DESENVOLVIMENTO DO APLICATIVO TO-DO-LIST

Brasil
2024

1. Implementação das Funcionalidades.
Desenvolver as telas e funcionalidades do aplicativo



- Tela de Login e cadastro de novo usuairo

O Código “auth_service.dart” implementa um serviço de autenticação em Flutter usando o Firebase Authentication. A classe **`AuthService`** gerencia cadastro, login, logout e monitora mudanças no estado de autenticação do usuário, notificando a interface via **`ChangeNotifier`**. Ela trata erros comuns como senha fraca ou e-mail inválido e mantém o estado do usuário atualizado em tempo real, facilitando a integração com aplicativos que requerem autenticação personalizada, como listas de tarefas. O código “social_login.dart” implementa a autenticação com o Google no Firebase em um aplicativo Flutter. Ele solicita o login do usuário via Google, obtém os tokens de autenticação, cria uma credencial Firebase e realiza o login, retornando as informações do usuário autenticado.

```dart
import 'package:firebase_auth/firebase_auth.dart';
import 'package:google_sign_in/google_sign_in.dart';

Future<UserCredential> signInWithGoogle() async {
  // Inicia o fluxo de autenticação
  final GoogleSignInAccount? googleUser = await GoogleSignIn().signIn();

  // Obtém os detalhes de autenticação
  final GoogleSignInAuthentication? googleAuth =
      await googleUser?.authentication;

  // Cria uma nova credencial
  final credential = GoogleAuthProvider.credential(
    accessToken: googleAuth?.accessToken,
    idToken: googleAuth?.idToken,
  );

  // Após o login bem-sucedido, retorna a credencial do usuário
  return await FirebaseAuth.instance.signInWithCredential(credential);
}

```

taskflow\lib\services\social_login.dart

```dart
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';

class AuthException implements Exception {
  String message;
  AuthException(this.message);
}

class AuthService extends ChangeNotifier {
  final FirebaseAuth _auth = FirebaseAuth.instance;
  late User? localUser = FirebaseAuth.instance.currentUser;
  bool isLoading = true;

  AuthService() {
    _authCheck();
  }

  _authCheck() {
    _auth.authStateChanges().listen((User? user) {
      localUser = (user == null) ? null : user;
      isLoading = false;
      notifyListeners();
    });
  }

  _getUser() {
    localUser;
    notifyListeners();
  }

  signup(String email, String password) async {
    try {
      await _auth.createUserWithEmailAndPassword(
          email: email, password: password);
      _getUser();
    } on FirebaseAuthException catch (e) {
      debugPrint("error-code: ${e.code}");
      if (e.code == 'weak-password') {
        throw AuthException('A senha é muito fraca!');
      } else if (e.code == 'email-already-in-use') {
        throw AuthException('Este email já está cadastrado.');
      }
    }
  }

  signin(String email, String password) async {
    try {
      await _auth.signInWithEmailAndPassword(email: email, password: password);
      _getUser();
    } on FirebaseAuthException catch (e) {
      debugPrint("error-code: ${e.code}");
      if (e.code == 'invalid-credential') {
        throw AuthException('Email ou senha incorreta');
      } else if (e.code == 'invalid-email') {
        throw AuthException('Email inválido.');
      }
    }
  }

  logout() async {
    await _auth.signOut();
    _getUser();
  }
}

```

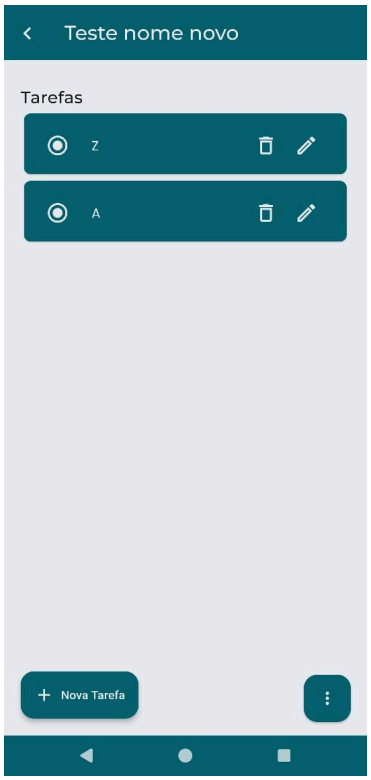
taskflow\lib\services\auth_service.dart

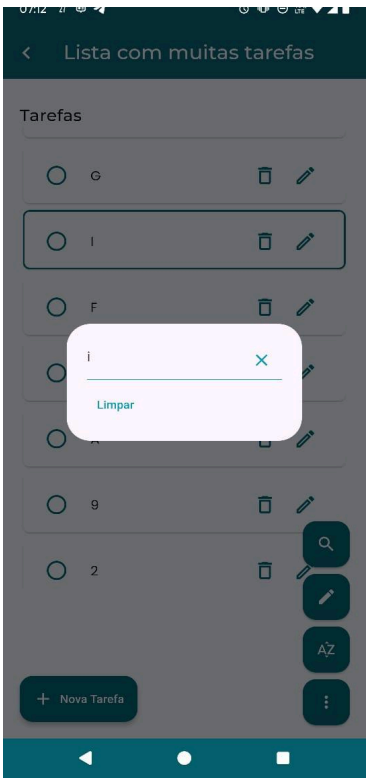
- Tela Principal (Lista de Tarefas)

O código”taskflow\lib\pages\list\list_page.dart” define a página principal de um aplicativo Flutter para gerenciar listas de tarefas com integração ao Firebase Firestore. Ele permite criar, editar, excluir, buscar e marcar listas como concluídas, além de personalizar o perfil do usuário com nome e foto. Utiliza **Showcase** para guiar novos usuários, oferece busca com destaque nos resultados e sincroniza alterações em tempo real com o Firebase. A interface é moderna, com design personalizado e funcionalidades práticas para organização pessoal.

```dart
class ListPageState extends State<ListPage> {
  final TextEditingController _searchController = TextEditingController();
  final TextEditingController _editUsernameController = TextEditingController();
  final GlobalKey _one = GlobalKey();
  final GlobalKey _two = GlobalKey();
  final GlobalKey _three = GlobalKey();
  final GlobalKey _four = GlobalKey();

  late List<Lists> tasklist;
  List<Lists> filteredLists = [];
  late ListRepository listRepository;
  final ScrollController _scrollController = ScrollController();
  User? user = FirebaseAuth.instance.currentUser;
  int? highlightedListIndex;
  String? userName;
  late File userProfilePic;
  bool hasProfilePicUploaded = false;
  String uniquekey = '';

  @override
  void initState() {
    super.initState();
    _checkFirstAccess();
    _loadUserNameWithRetry();
    _loadUserProfilePic();
  }

  @override
  void didChangeDependencies() {
    super.didChangeDependencies();
    listRepository = Provider.of<ListRepository>(context);
    listRepository.addListener(_updateTaskList);
    _updateTaskList();
    if (_searchController.text.isEmpty) {
      _clearSearch();
    }
  }

  @override
```

taskflow\lib\pages\list\list_page.dart





- Tela de Adicionar/Editar Tarefa

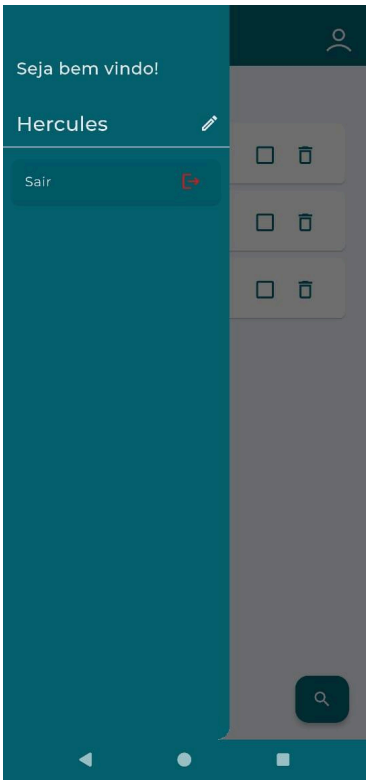
Os códigos “task_add_page.dart” e “task_edit_page.dart” implementam as páginas de **adição** e **edição** do aplicativo. A página de adição permite criar novas tarefas associadas a uma lista, com campos para o nome e data de conclusão, validados antes de salvar. Já a página de edição permite modificar o nome e a data de uma tarefa existente, identificada por um ID, carregando as informações atuais e atualizando o repositório após as alterações. Ambas oferecem uma interface intuitiva, feedback visual, e seguem o design consistente do aplicativo, utilizando validação de dados e integração com o repositório de tarefas.

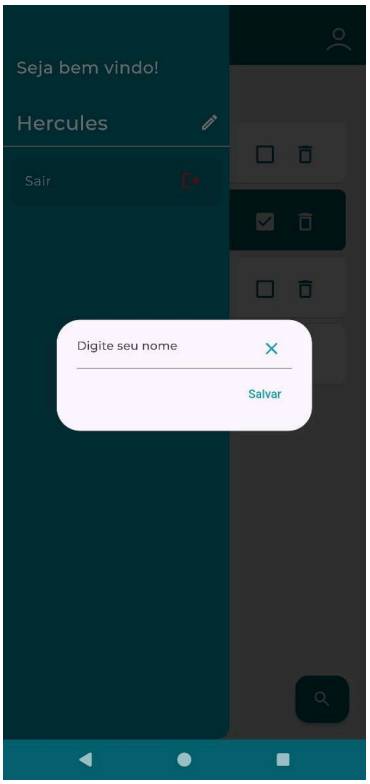
```dart
class TaskAddPage extends StatefulWidget {
  static String tag = 'task_add_page';
  final String listId;

  const TaskAddPage({super.key, required this.listId});

  @override
  TaskAddPageState createState() => TaskAddPageState();
}

class TaskAddPageState extends State<TaskAddPage> {
  final DateFormat _dateFormat = DateFormat('dd/MM/yyyy', 'pt_BR');
  final _dateController = MaskedTextController(mask: '00/00/0000');
  final TextEditingController _taskNameController = TextEditingController();
  final GlobalKey<FormState> _formNameKey = GlobalKey<FormState>();
  final GlobalKey<FormState> _formDateKey = GlobalKey<FormState>();
  late TasksRepository tasksRepository;

```

taskflow\lib\pages\task\task_add_page.dart

```dart

class TaskEditPage extends StatefulWidget {
  static String tag = 'task_edit_page';
  final String taskId;

  const TaskEditPage({super.key, required this.taskId});

  @override
  TaskEditPageState createState() => TaskEditPageState();
}

class TaskEditPageState extends State<TaskEditPage> {
  final DateFormat _dateFormat = DateFormat('dd/MM/yyyy', 'pt_BR');
  final _dateController = MaskedTextController(mask: '00/00/0000');
  final TextEditingController _taskNameController = TextEditingController();
  final GlobalKey<FormState> _formNameKey = GlobalKey<FormState>();
  final GlobalKey<FormState> _formDateKey = GlobalKey<FormState>();
  late TasksRepository tasksRepository;

```

taskflow\lib\pages\task\task_edit_page.dart





- Tela de Configurações

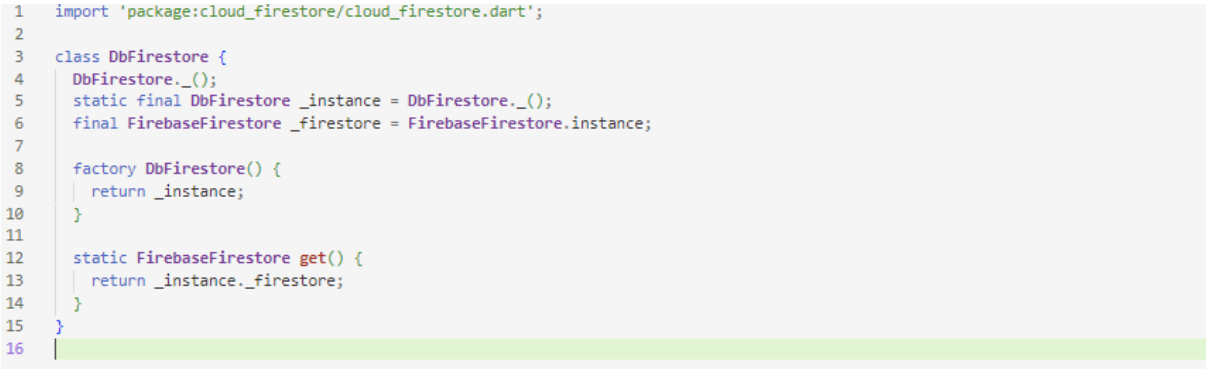
O **drawer** no código “list_page.dart”, atualmente configurado para exibir e permitir a edição do nome do usuário, oferece uma estrutura flexível que pode ser facilmente expandida para incluir mais funcionalidades. Além do gerenciamento básico de perfil, ele pode suportar configurações adicionais, como alternar entre temas (claro/escuro), ativar ou desativar notificações, selecionar idiomas, sincronizar dados, acessar páginas de ajuda e suporte, visualizar informações sobre o aplicativo, alterar senha ou gerenciar permissões. Apesar de inicialmente focado na edição do nome, o **drawer** tem potencial para se tornar um centro completo de configurações e personalização, melhorando a experiência do usuário.

```
849       drawer: SizedBox(
850         width: 270,
851         child: Drawer(
852           child: Container(
853             color: AppColors.primaryGreenColor,
854             child: ListView(
855               children: <Widget>[
856                 UserAccountsDrawerHeader(
857                   accountName: const SizedBox(
858                     height: 70,
859                     child: Text(
860                       'Seja bem vindo!',
861                       style: TextStyle(
862                         color: AppColors.primaryWhiteColor,
863                         fontFamily: AppFonts.montserrat,
864                         fontSize: 20.0,
865                         fontWeight: FontWeight.w500,
866                       ),
867                     ),
868                   ),
869                   accountEmail: Row(
870                     textDirection: TextDirection.ltr,
871                     children: <Widget>[
872                       Text(
873                         userName?.isNotEmpty == true
874                             ? userName!
875                             : 'Insira seu nome',
876                         style: const TextStyle(
877                           color: AppColors.primaryWhiteColor,
878                           fontFamily: AppFonts.montserrat,
879                           fontSize: 24.0,
880                           fontWeight: FontWeight.w500,
881                         ),
882                       ),
883                       const Spacer(),
```

taskflow\lib\pages\list\list_page.dart

Banco de dados

O Firebase foi escolhido para este aplicativo devido às suas diversas vantagens que atendem às necessidades do projeto. Ele oferece um **banco de dados em tempo real** e uma **integração simples com o Firestore**, permitindo sincronização de dados rápida e confiável entre os usuários. Além disso, suas **ferramentas integradas**, como autenticação, armazenamento e notificações push, reduzem a complexidade do desenvolvimento e garantem escalabilidade. A facilidade de configuração e suporte a múltiplas plataformas fazem do Firebase uma solução ideal para aplicativos móveis que requerem desempenho, segurança e flexibilidade.

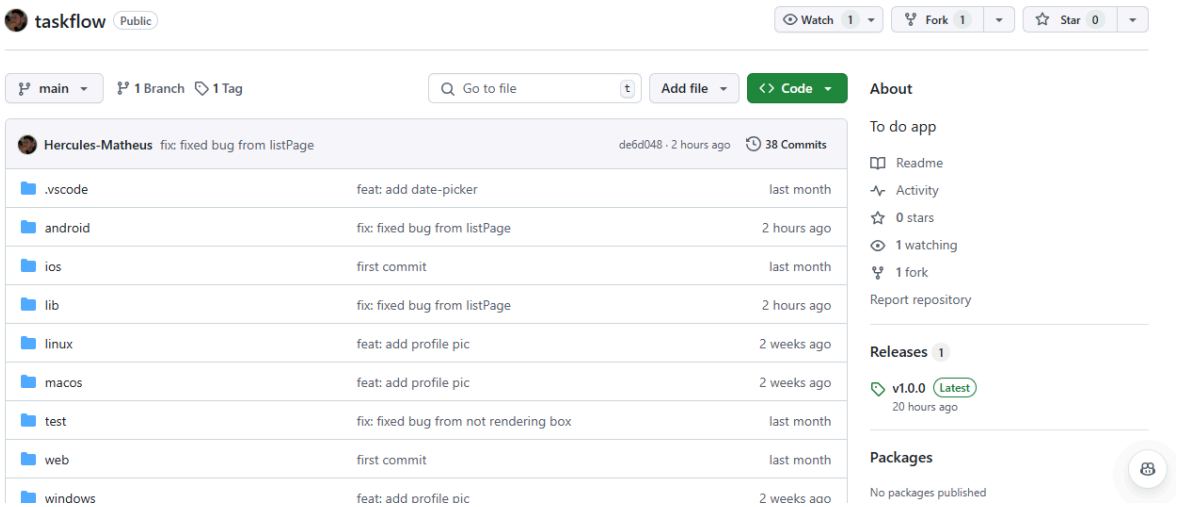


```dart
import 'package:cloud_firestore/cloud_firestore.dart';

class DbFirestore {
  DbFirestore._();
  static final DbFirestore _instance = DbFirestore._();
  final FirebaseFirestore _firestore = FirebaseFirestore.instance;

  factory DbFirestore() {
    return _instance;
  }

  static FirebaseFirestore get() {
    return _instance._firestore;
  }
}
```

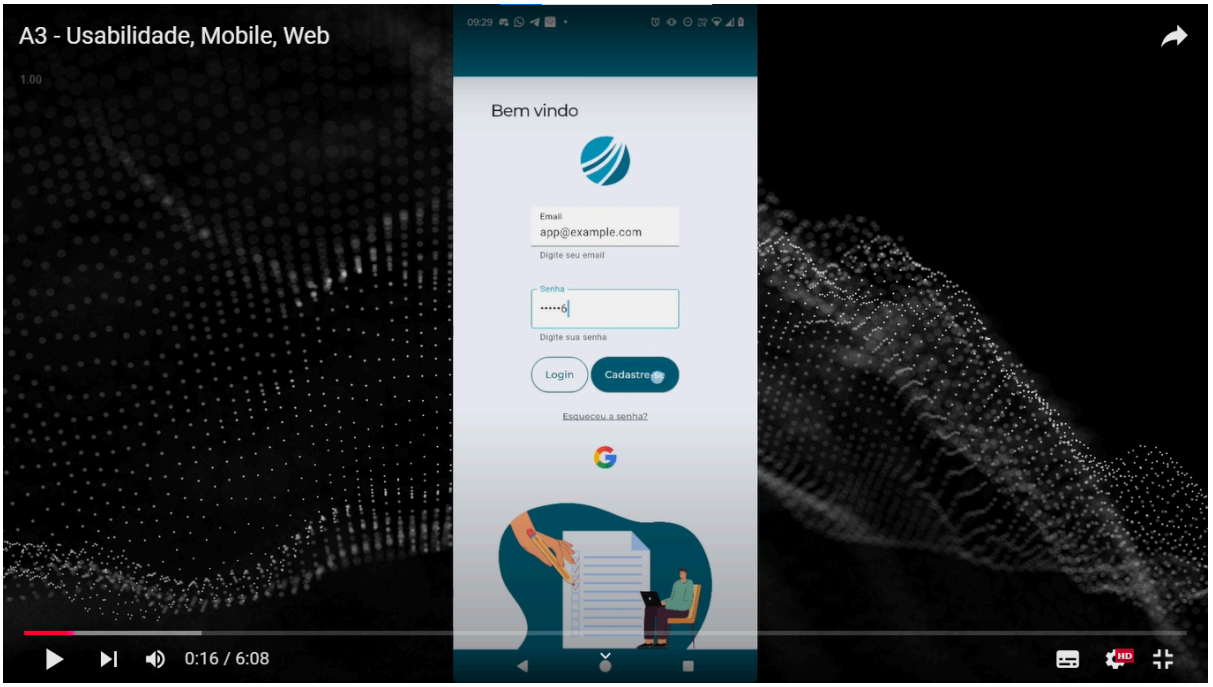
taskflow\lib\database\db_firestore.dart

Links do projeto

link Github:
https://github.com/Hercules-Matheus/taskflow



link Video:
https://youtu.be/En_KTMqq4Kw



Referências

1. **Documentação Oficial do Flutter**
   - Flutter Dev. *Build apps for any screen*. Disponível em: https://flutter.dev/
   - Acesso em: 29 nov. 2024.
2. **Firebase para Flutter**
   - Firebase. *Add Firebase to your Flutter app*. Disponível em: https://firebase.google.com/docs/flutter/setup
   - Acesso em: 29 nov. 2024.
3. **Singleton Design Pattern no Flutter**
   - Dart Dev. *Using the Singleton pattern in Dart*. Disponível em: https://dart.dev/guides/language/effective-dart/design
   - Acesso em: 29 nov. 2024.
4. **Gerenciamento de Estado com Provider**
   - Flutter Dev. *Simple app state management*. Disponível em: https://flutter.dev/docs/development/data-and-backend/state-mgmt/simple
   - Acesso em: 29 nov. 2024.
5. **Heurísticas de Usabilidade de Nielsen**
   - UX Design. *10 heurísticas de Nielsen para o design de interface*. Disponível em: https://brasil.uxdesign.cc/
   - Acesso em: 29 nov. 2024.
6. **Autenticação no Firebase**
   - Firebase. *Authenticate with Firebase using Password-Based Accounts*. Disponível em: https://firebase.google.com/docs/auth/web/password-auth
   - Acesso em: 29 nov. 2024.
7. **Integração de Autenticação com Google no Firebase**
   - Firebase. *Authenticate with Firebase using Google Sign-In*. Disponível em: https://firebase.google.com/docs/auth/flutter/google-signin
   - Acesso em: 29 nov. 2024.
8. **Gerenciamento de Tarefas no Firestore**
   - Firebase. *Cloud Firestore*. Disponível em: https://firebase.google.com/docs/firestore
   - Acesso em: 29 nov. 2024.
9. **Uso do Drawer no Flutter**
   - Flutter Dev. *Material Design Drawer*. Disponível em: https://api.flutter.dev/flutter/material/Drawer-class.html
   - Acesso em: 29 nov. 2024.
10. **Playlist YouTube Flutter na Prática**
    - Flutter na Prática. Disponível em: https://www.youtube.com/@drantunes
    - Acesso em: 29 nov. 2024.