**UNIVERSIDADE NOVE DE JULHO**

**DIRETORIA DOS CURSOS DE INFORMÁTICA**

**Alexandro Xavier de Oliveira - RA 920124023**

**Anderson Henrique da Silva - RA 920116006**

**Bryan Matheus Felicíssimo dos Santos - RA 920123128**

**Gustavo Oliveira Felix - RA 920114381**

**Lucas Vinicius De Sousa Lara - RA 920123986**

**Matheus Lucas Januário de Matos - RA 920125859**

**Sandro Henrique Gomes - RA 920113843**

**Vanessa Aparecida de Jesus Antônio - RA 920116318**

**Vinícius Ryan Cruz da Silva - RA 920122403**

**PROJETO PRÁTICO EM SISTEMAS**

*IFS - Intelligent Facilities And Services*

**SÃO PAULO**

**2021**

**Alexandro Xavier de Oliveira - RA 920124023**

**Anderson Henrique da Silva - RA 920116006**

**Bryan Matheus Felicíssimo dos Santos - RA 920123128**

**Gustavo Oliveira Felix - RA 920114381**

**Lucas Vinicius De Sousa Lara - RA 920123986**

**Matheus Lucas Januário de Matos - RA 920125859**

**Sandro Henrique Gomes - RA 920113843**

**Vanessa Aparecida de Jesus Antônio - RA 920116318**

**Vinícius Ryan Cruz da Silva - RA 920122403**

**PROJETO PRÁTICO EM SISTEMAS**

*IFS - Intelligent Facilities And Services*

Trabalho apresentado à Universidade Nove de Julho, UNINOVE, em cumprimento parcial às exigências da disciplina de Projeto Prático em Sistemas, sob orientação do Prof. **Eliane Aparecida Peixoto Favilla**

**SÃO PAULO**

**2021**

ROTEIRO DO PROJETO

Para o desenvolvimento do Projeto é proposto a criação de um Sistema Móvel (APP Android), utilizando os conceitos aprendidos nas disciplinas de Sistemas Móveis (Android).

Os conceitos desenvolvidos nas disciplinas devem ser empregados em sua completude, ou seja, devem estar presentes em todas as etapas do projeto.

O projeto não restringe a utilização de outras tecnologias, mesmo que não tenham sido abordadas no curso.

Resumo

O propósito do App iBurguer é proporcionar mais eficiência para hamburguerias, utilizando de alguns métodos de controle de estoque,vendas e cadastro de usuários.

**Palavras chave:**

**iBurguer,App, Software, Javascript, Node.js, PostgreSQL,Arquitetura do software, controle, estoque, cardápio, wireframe, organograma, conclusão, desenvolvimento, metodologia, design, flutter, Windows.**

Lista de Figuras

[Figura 1 — Organograma. 11](#_Toc20211870)

[Figura 2 — Wireframe. 15](#_Toc20211871)

[Figura 3 — Diagrama de classes. 24](#_Toc20211870)

[Figura 4 — Arquitetura de Software. 46](#_Toc20211870)

Sumário

[ROTEIRO DO PROJETO 3](#_Toc20211841)

[Resumo 4](#_Toc20211842)

[Lista de Figuras 5](#_Toc20211843)

[1. Introdução 7](#_Toc20211844)

[1.1. Motivações e Objetivo 7](#_Toc20211845)

[1.2. Descrição do Produto 8](#_Toc20211846)

[1.3. Premissas 8](#_Toc20211847)

[1.4. Recursos 8](#_Toc20211848)

[1.5. Definição do Negócio 9](#_Toc20211849)

[1.6. Definição da Equipe 9](#_Toc20211850)

[1.6.1. organograma 11](#_Toc20211851)

[2. Descrição do Sistema 12](#_Toc20211852)

[2.1. Descrição Detalhada das Partes que Compõe o Sistema 12](#_Toc20211853)

[2.1.1. Página Inicial 13](#_Toc20211854)

[2.1.2. Pesquisa 13](#_Toc20211855)

[2.2. Requisitos Funcionais 14](#_Toc20211856)

[2.3. Wireframes 15](#_Toc20211857)

[3. Modelagem UML 24](#_Toc20211858)

[3.1. Diagramas de Classes 24](#_Toc20211859)

[4. Modelagem do Banco de Dados 25](#_Toc20211860)

[4.1. Diagrama E-R 25](#_Toc20211861)

[4.2. Implementação Física 25](#_Toc20211862)

[5. Metodologia 28](#_Toc20211863)

[5.1. Desenvolvimento 28](#_Toc20211864)

[6. Arquitetura de Software 46](#_Toc20211865)

[6.1. Desenvolvimento 46](#_Toc20211866)

[7. Ferramentas Utilizadas 47](#_Toc20211867)

[8. Conclusão 49](#_Toc20211868)

[9. Bibliografia 51](#_Toc20211869)

1. Introdução

A IFS – Intelligent Facilities And Services, busca a automação dos processos de hamburguerias através do APP IBurger, com o propósito de oferecer maior controle sobre ganhos, recursos e funcionamento de sua hamburgueria.

O banco de dados está desenvolvido com a linguagem de programação Flutter para gerenciar desde o orçamento até os ingredientes e demais itens necessários para a comercialização dos produtos e funcionamento da hamburgueria.

A possibilidade de donos de hamburguerias terem um APP que facilite essa dinâmica entre recursos em falta com fornecedores é muito baixa. Com isso, a IFS fornece essa interação entre lojista, fornecedor e funcionários.

O APP irá mudar a maneira que os donos de hamburgueria administram seu negócio, de uma forma simples, colocando os lucros semanais ou mensais, os salários dos funcionários, gastos com produtos de estoque, reposição de estoque e manutenção.

* 1. Motivações e Objetivo

Motivados com a falta de softwares completos e dinâmicos focados em hamburguerias, a IFS – Intelligent Facilities And Services desenvolveu o APP IBurger, focando em seu desempenho, custo-benefício e tendo como objetivo auxiliar o funcionamento de hamburguerias de pequeno a médio porte. O IBurguer disponibiliza recursos que tornam dinâmicos os processos cotidianos, como o gerenciamento de recursos, ganhos e funcionalidades gerais de hamburguerias, assim buscando ganhar espaço em um mercado carente de ferramentas inovadoras.

* 1. Descrição do Produto

A IFS – Intelligent Facilities And Services, busca a automação de hamburguerias com aplicativos mobile, o mesmo permitirá que todos os funcionários consigam realizar o controle de reservas de mesas e cadastramento dos clientes, este aplicativo também permitirá que seja realizado todos os pedidos e o controle de caixa, a partir do momento em que o cliente realiza um pedido haverá uma contagem do caixa o valor do lucro e a quantidade de ingrediente está saindo, assim podendo ter o controle do fluxo de caixa e de estoque ao mesmo tempo em que o pedido está sendo realizado, e ao final do dia o proprietário terá a possibilidade de ver todos esses dados para se planejar e sempre ter o controle de seu negócio em todas as áreas.

* 1. Premissas

O aplicativo será desenvolvido em no máximo 6 meses;

O aplicativo será construído em Flutter;

O aplicativo deverá consumir uma base de dados;

Será disponibilizado o gerenciamentode recursos para hamburguerias.

* 1. Recursos ?

Para desenvolvermos nosso aplicativo, precisaremos de um escritório com um computador, mouse, teclado, monitor, uma mesa e uma cadeira para cada integrante da empresa. Por conta da pandemia vigente, cada sócio da empresa irá trabalhar em Home Office. Em termos de software, utilizaremos o programa Flutter pra criar a compilação do aplicativo e o SQL Server para gerenciar o banco de dados de nosso projeto. A escolha do SO (Sistema operacional) foi o Windows, por conta da facilidade de uso e alta compatibilidade com softwares.

* 1. Definição do Negócio

O aplicativo tem por objetivo atender á donos e funcinarios das hambúrguerias de pequeno e médio porte. E tornar mais dinâmico os processo do cotidiano, gerenciamento, ganhos e funcionalidades.

* 1. Definição da Equipe

Alexandro Xavier de Oliveira

RA: 920124023

alexavierdeoliveiraa@gmail.com

Anderson Henrique da Silva

RA: 920116006

a.henrique@uni9.edu.br

Bryan Matheus Felicíssimo dos Santos

RA: 920123128

bryan.matheusbmf@uni9.edu.br

Gustavo Oliveira Felix

RA: 920114381

gufelix234@uni9.edu.br

Nome: Lucas Vinicius De Sousa Lara

RA: 920123986

lucasfabiocosta@uni9.edu.br

Matheus Lucas Januário de Matos

RA: 920125859

rosangella17.jmattos@uni9.edu.br

Sandro Henrique Gomes

RA:920113843

henriquegomes@uni9.edu.br

Vanessa Aparecida de Jesus Antônio

RA. 920116318

vanessa1984@uni9.edu.br

Vinícius Ryan Cruz da Silva

RA: 920122403

viniciusryan1032@uni9.edu.br

1.6.1.organograma

Lucas como Coordenador será responsável por criar e acompanhar o cronograma do projeto, distribuindo as tarefas para os profissionais e auxiliará o grupo de Desenvolvimento. Vanessa e Matheus serão responsáveis por elaborar o desenho das interfaces do app, primando pela usabilidade, elemento fundamental para o sucesso do aplicativo.

Já o Matheus irá aplicar o layout projetado anteriormente. As técnicas são parecidas com as de um site, mas usando outro tipo de tecnologia.

Henrique será responsável por compreender a necessidade de negócio do cliente e especificar por escrito o que precisa ser feito no projeto. Gustavo, Vinicius Ryan, irão analisar as necessidades do projeto e definir a arquitetura técnica que melhor se encaixa noy projeto.

Bryan, Alexandro transformarão as especificações de negócio do aplicativo em código, seguindo as diretrizes técnicas do arquiteto e análise funcional do analista de sistemas. O código fonte faz a conexão com banco de dados e a camada visual, para leitura, gravação e exposição das informações. Anderson Henrique será responsável por definir a arquitetura do banco de dados e apoiar na criação dos comandos para tratamento das rotinas mais complexas de manipulação de dados. Bruna fará a validação do aplicativo, ou seja, se a etapa de codificação cumpriu o que foi solicitado na especificação do Analista de Sistemas e se não há erros (bugs) no app.

1. Descrição do Sistema

Controle de Vendas

Para ter total controle sobre as vendas, o aplicativo oferece recursos como: controle de pedidos, controle de caixa, vendas por produto, fornecedor e marca, relatório de faturamento, acompanhamento de vendas em tempo real e muito mais!

Impressão de Comanda na Cozinha

Você pode imprimir uma comanda de pedido em mais de uma impressora na cozinha.

Venda por combo

Controle as promoções de venda por combo com facilidade. Por exemplo: crie um combo de venda vinculada de hambúrguer, batata e refrigerante e assim simplifique e agilize a venda e o controle de estoque.

Controle de Caixa

Abertura e fechamento de caixa, controle de caixa por turno/funcionário, entrada (suprimento) e saída (sangria) de dinheiro do caixa, relatório de caixa por e-mail e controle de caixa por meio de pagamento são alguns dos recursos do controle total de seu caixa.

* 1. Descrição Detalhada das Partes que Compõe o Sistema

O sistema é composto por três partes. A primeira sendo o back-end da aplicação, uma API (Application Programming Interface) que fará a ponte entre o banco de dados e o aplicativo e será construída com NodeJS um runtime para a linguagem Javascript. E a segunda parte será o aplicativo móvel, construído em Flutter, um framework da linguagem Dart. Além disso, a ultima parte será o banco de dados relacional e constuido em PostgreSQL.

* + 1. Página Inicial

Logo da empresa

Login dos funcionários e proprietário.

Botão de cadastro

* + 1. Pesquisa

Logo da empresa

Login dos funcionários e proprietário.

Tela de cadastro

Ícone de cardápio

Ícone de comanda

Ícone estoque

Ícone finanças

Ícone configuração

Cardápios lanches e bebidas

Imagem do lanche e bebidas adicionados ao cardápio.

Registro da mesa

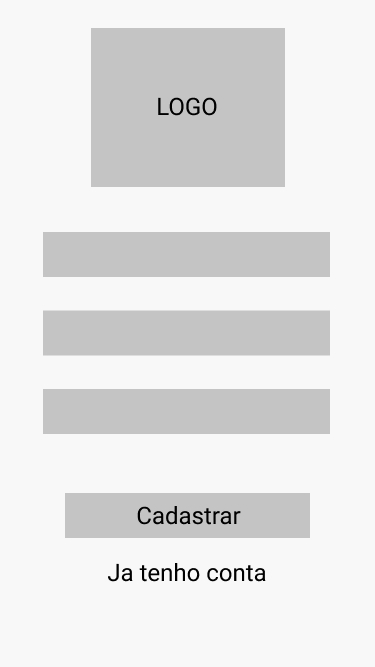
Estoque final após pedido realizado.

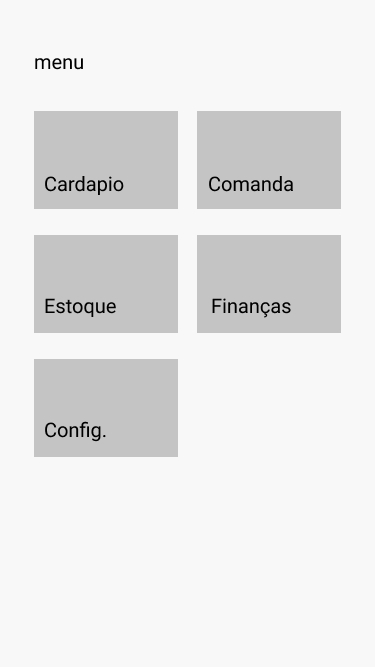
* 1. Requisitos Funcionais
* O Sistema deve cadastrar todos os clientes.
* Deverá ter acesso ao banco de dados para manter o controle dos clientes.
* Ter acesso ao banco de dados para manter a lista de lanches e de ingredientes atualizada.
* A cada pedido realizado o sistema deverá realizar o controle do estoque retirando a mesma quantidade de ingredientes necessários para a produção dos lanches.
* Ao Final do dia deverá realizar um levantamento de quantos ingredientes tem no estoque, quantidade de clientes, quantos lanches foram vendidos e o lucro.

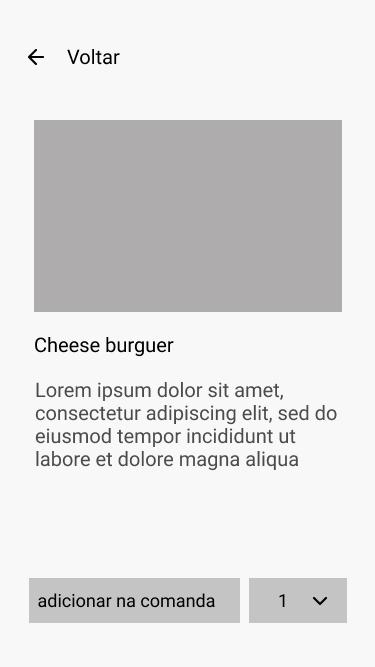
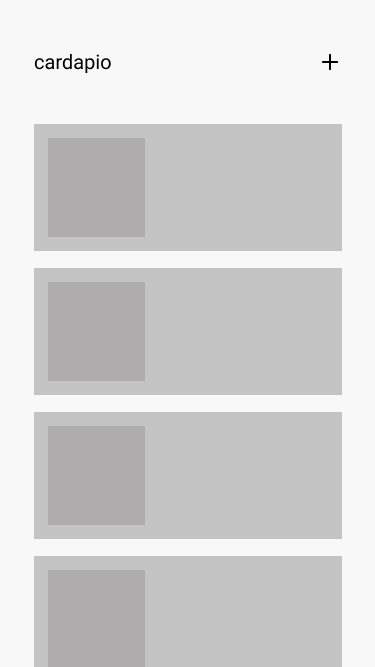
* 1. Wireframes

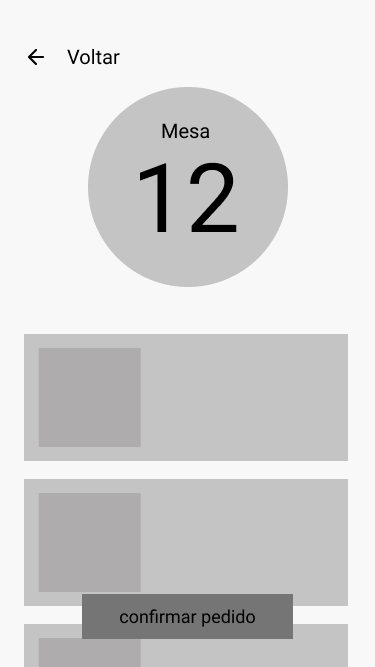


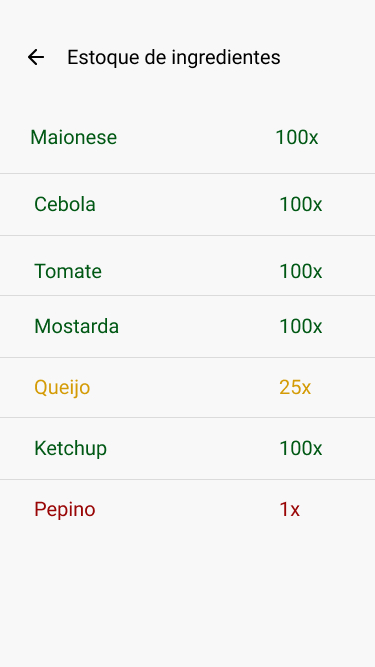












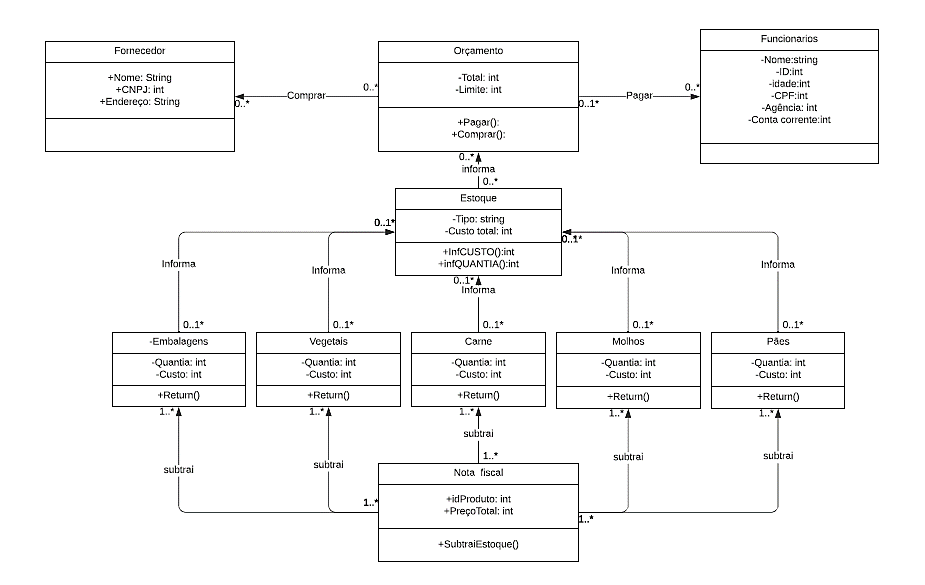
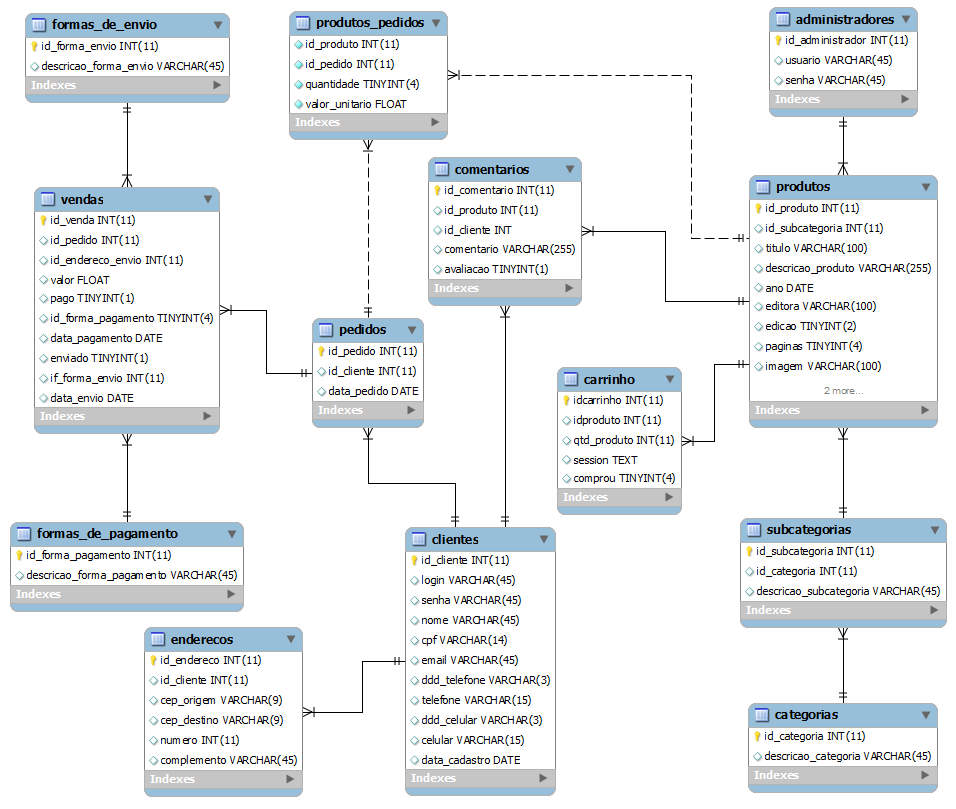
1. Modelagem UML
   1. Diagramas de Classes
2. Modelagem do Banco de Dados
   1. Diagrama E-R

Figura 5 — Diagrama Entidade-Relacionamento.



Fonte: Os Autores.

* 1. Implementação Física

CREATE TABLE EMPRESA (

ID\_EMPRESA NUMERIC PRIMARY KEY,

RAZAO\_SOCIAL VARCHAR(50),

NOME\_FANTASIA VARCHAR(50),

CNPJ NUMERIC(25)

);

CREATE TABLE FUNCIONARIO (

ID\_FUNCIONARIO NUMERIC PRIMARY KEY,

NOME\_FUNC VARCHAR(50),

DATA\_NASC DATE,

CPF NUMERIC(14),

DATA\_ADM DATE,

DATA\_DEM DATE,

CARGO VARCHAR(50),

SALARIO REAL

);

CREATE TABLE FORNECEDOR (

ID\_FORNECEDOR NUMERIC PRIMARY KEY,

NOME\_FORNECEDOR VARCHAR(50),

CNPJ\_FORNECEDOR NUMERIC(25),

ENDERECO\_FORNECEDOR VARCHAR(50),

VALOR\_FORNECEDOR REAL

);

CREATE TABLE PRODUTO (

ID\_PRODUTOS NUMERIC PRIMARY KEY,

TIPO\_PRODUTO VARCHAR(50),

VALOR\_PRODUTO REAL

);

CREATE TABLE MANUTENCAO (

ID\_MANUTENCAO NUMERIC PRIMARY KEY,

TIPO\_MANUTENCAO VARCHAR(50),

DATA\_MANUTENCAO DATE,

VALOR\_MANUTENCAO REAL

);

CREATE TABLE DIVERSOS (

ID\_DIVERSOS NUMERIC PRIMARY KEY,

TIPO\_DIVERSO VARCHAR(50),

VALOR\_DIVERSO REAL

);

5.Metodologia

5.1 Desenvolvimento

**Home page**

import 'package:flutter/material.dart';

class HomePage extends StatelessWidget {

@override

Widget build(BuildContext context) {

return Scaffold(

body:

GridView.count(

primary: false,

padding: const EdgeInsets.only(

top: 10,

left: 40,

right: 40,

),

crossAxisSpacing: 10,

mainAxisSpacing: 10,

crossAxisCount: 2,

children: <Widget>[

Container(

padding: const EdgeInsets.only(

top: 10,

left: 8,

right: 8,

),

margin: const EdgeInsets.only(

top: 70,

),

child: Text(

'Menu',

style: const TextStyle(

color: Colors.black87,

fontSize: 20,

fontWeight: FontWeight.bold,

),

)

),

SizedBox(),

Container(

padding: const EdgeInsets.only(

top: 70,

left: 8,

right: 8,

),

child: FlatButton(

child: Row(

mainAxisAlignment: MainAxisAlignment.spaceBetween,

children: <Widget>[

Text(

"Cardapio",

style: TextStyle(

fontWeight: FontWeight.bold,

color: Colors.white,

fontSize: 20,

),

textAlign: TextAlign.left,

),

],

),

onPressed: () {

Navigator.push(

context,

MaterialPageRoute(builder: (context) => HomePage()),

);

},

),

color: Colors.black87,

),

Container(

padding: const EdgeInsets.only(

top: 70,

left: 8,

right: 8,

),

child: FlatButton(

child: Row(

mainAxisAlignment: MainAxisAlignment.spaceBetween,

children: <Widget>[

Text(

"Comanda",

style: TextStyle(

fontWeight: FontWeight.bold,

color: Colors.white,

fontSize: 20,

),

textAlign: TextAlign.left,

),

],

),

onPressed: () {

Navigator.push(

context,

MaterialPageRoute(builder: (context) => HomePage()),

);

},

),

color: Colors.black87,

),

Container(

padding: const EdgeInsets.only(

top: 70,

left: 8,

right: 8,

),

child: FlatButton(

child: Row(

mainAxisAlignment: MainAxisAlignment.spaceBetween,

children: <Widget>[

Text(

"Estoque",

style: TextStyle(

fontWeight: FontWeight.bold,

color: Colors.white,

fontSize: 20,

),

textAlign: TextAlign.left,

),

],

),

onPressed: () {

Navigator.push(

context,

MaterialPageRoute(builder: (context) => HomePage()),

);

},

),

color: Colors.black87,

),

Container(

padding: const EdgeInsets.only(

top: 70,

left: 8,

right: 8,

),

child: FlatButton(

child: Row(

mainAxisAlignment: MainAxisAlignment.spaceBetween,

children: <Widget>[

Text(

"Finanças",

style: TextStyle(

fontWeight: FontWeight.bold,

color: Colors.white,

fontSize: 20,

),

textAlign: TextAlign.left,

),

],

),

onPressed: () {

Navigator.push(

context,

MaterialPageRoute(builder: (context) => HomePage()),

);

},

),

color: Colors.black87,

),

Container(

padding: const EdgeInsets.only(

top: 70,

left: 8,

right: 8,

),

child: FlatButton(

child: Row(

mainAxisAlignment: MainAxisAlignment.spaceBetween,

children: <Widget>[

Text(

"Config",

style: TextStyle(

fontWeight: FontWeight.bold,

color: Colors.white,

fontSize: 20,

),

textAlign: TextAlign.left,

),

],

),

onPressed: () {

Navigator.push(

context,

MaterialPageRoute(builder: (context) => HomePage()),

);

},

),

color: Colors.black87,

),

],

)

);

}

}

**Main**

import 'package:flutter/material.dart';

import 'package:app\_iburger/pages/login\_page.dart';

void main() {

runApp(MyApp());

}

class MyApp extends StatelessWidget {

@override

Widget build(BuildContext context) {

return MaterialApp(

title: 'IBurger',

debugShowCheckedModeBanner: false,

theme: ThemeData(

primarySwatch: Colors.grey,

),

home: LoginPage(),

);

}

}

**Login page**

import 'package:app\_iburger/pages/home\_page.dart';

import 'package:flutter/material.dart';

class LoginPage extends StatelessWidget{

@override

Widget build(BuildContext context){

return Scaffold(

body: Container(

padding: EdgeInsets.only(

top: 140,

left: 40,

right: 40,

),

color: Colors.white,

child: ListView(

children: <Widget>[

SizedBox(

width: 128,

height: 128,

child: Image.asset("assets/logo.png"),

),

SizedBox(

height: 50,

),

TextFormField(

//autofocus: true,

keyboardType: TextInputType.text,

decoration: InputDecoration(

labelText: "Usuário",

labelStyle: TextStyle(

color: Colors.black87,

fontWeight: FontWeight.w400,

fontSize: 20,

),

),

style: TextStyle(

fontSize: 20,

),

),

SizedBox(

height: 10,

),

TextFormField(

//autofocus: true,

keyboardType: TextInputType.text,

obscureText: true,

decoration: InputDecoration(

labelText: "Senha",

labelStyle: TextStyle(

color: Colors.black87,

fontWeight: FontWeight.w400,

fontSize: 20,

)

),

style: TextStyle(

fontSize: 20,

),

),

SizedBox(

height: 40,

),

Container(

height: 60,

alignment: Alignment.centerLeft,

decoration: BoxDecoration(

gradient: LinearGradient(

begin: Alignment.topLeft,

end: Alignment.bottomRight,

stops: [0.3, 1],

colors: [

//Color(0xFF303030),

//Color(0xFF757575),

Color(0xFF757575),

Color(0xFF9E9E9E),

],

),

borderRadius: BorderRadius.all(

Radius.circular(5),

)

),

child: SizedBox.expand(

child: FlatButton(

child: Row(

mainAxisAlignment: MainAxisAlignment.spaceBetween,

children: <Widget>[

Text(

"Entrar",

style: TextStyle(

fontWeight: FontWeight.bold,

color: Colors.black87,

fontSize: 20,

),

textAlign: TextAlign.left,

),

Container(

child: SizedBox(

child: Image.asset("assets/user.png"),

height: 40,

width: 40,

),

)

],

),

onPressed: () {

Navigator.push(

context,

MaterialPageRoute(builder: (context) => HomePage()),

);

},

),

),

),

],

),

),

);

}

}

**Cardápio e estoque**

A)Para começarmos o código do cardápio e do estoque, primeiro declaramos as variáveis.

B)Desenvolvemos o design do código, criando os widgets, layouts e etc.

C)Criamos as classes.

D)Finalizamos com a criação das funções.

import 'package:flutter/material.dart';

void main() => runApp(MyApp());

class MyApp extends StatelessWidget {

final TextEditingController \_controladorNome = TextEditingController();

final TextEditingController \_controladorQuantidade = TextEditingController();

final TextEditingController \_controladorValor = TextEditingController();

@override

Widget build(BuildContext context) {

return MaterialApp(

home: Scaffold(

appBar: AppBar(

title: Text('Estoque'),

),

body: Padding(

padding: const EdgeInsets.all(16.0),

child: Column(

children: <Widget>[

TextField(

controller: \_controladorNome,

decoration: InputDecoration(labelText: 'Nome'),

),

Padding(

padding: const EdgeInsets.only(top: 16.0),

child: TextField(

controller: \_controladorQuantidade,

decoration: InputDecoration(labelText: 'Quantidade'),

keyboardType: TextInputType.number,

),

),

Padding(

padding: const EdgeInsets.only(top: 16.0),

child: TextField(

controller: \_controladorValor,

decoration: InputDecoration(labelText: 'Valor'),

keyboardType: TextInputType.number,

),

),

Padding(

padding: const EdgeInsets.only(top: 16.0),

child: RaisedButton(

child: Text('Cadastrar'),

onPressed: () {

final String nome = \_controladorNome.text;

final int quantidade =

int.tryParse(\_controladorQuantidade.text);

final double valor =

double.tryParse(\_controladorValor.text);

final Produto produtoNovo =

Produto(nome, quantidade, valor);

print(produtoNovo);

},

),

),

],

),

),

),

);

}

}

class Produto {

final String nome;

final int quantidade;

final double valor;

Produto(

this.nome,

this.quantidade,

this.valor,

);

@override

String toString() {

return 'Produto{nome: $nome, quantidade: $quantidade, valor: $valor}';

}

}

1. Arquitetura de Software
   1. Desenvolvimento

Gráfico

Descrição gerada automaticamente

O sistema foi construído 3 camadas no modelo, onde:

- Aplicativo:

Desenvolvido em Flutter, é a parte principal do sistema. Se comunica com a API,

através de requisições e respostas via HTTP.

- API:

Desenvolvida em Typescript/ Node.js é o componente para comunicação entre o aplicativo e o banco de dados, utilizando JSON.

- Banco de Dados:

Construído em PostgreSQL, é onde ficará armazenado todos os dados da aplicação e se comunicando diretamente com a API através de requisições.

7.Ferramentas Utilizadas

As ferramentas utilizadas no desenvolvimento do projeto foram selecionadas com o objetivo de facilitar e agilizar a produção de cada etapa do trabalho, sendo separadas como ferramentas de produção e comunicação.

Ferramentas de produção:

* Flutter: Framework da linguagem de programação Dart utilizado para desenvolver o software mobile IBurger.
* JavaScript: Linguagem de programação utilizada para desenvolver a API responsável pela comunicação do aplicativo IBurger com o banco de daods.
* Node.js: Software que executa códigos JavaScript no back-end que foi utilizado para desenvolver a API.
* Yarn: Gerenciador de pacotes JavaScript.
* PostgreSQL: Gerenciador de banco de dados.
* Docker: Plataforma para criação de ambientes isolados através de containers.
* TypeORM: Ferramenta utilizada para facilitar a consulta de manipulação dados do banco de dados por meio de um paradigma orentado a objetos.
* Prisma: Ferramenta utilizada para facilitar a consulta de manipulação dados do banco de dados por meio de um paradigma orentado a objetos.
* Android Studio: Plataforma utilizada para o desenvolvimento do aplicativo IBurger.
* Visual Studio Code: Plataforma utilizada para o desenvolvimento do aplicativo IBurger.
* Github: Ferramenta online utilizada para controle de versão.
* LucidChart: Ferramenta online para criação de gráficos utilizada no desenvolvimento do diagrama de classes.
* Figma: Ferramenta online utilizada para o desenvolvimento dos wireframes e design do aplicativo.

Ferramentas de comunicação:

* Whatsapp: Aplicativo multiplataforma utilizado para comunicação por meio de mensagens.
* Discord: Aplicativo multiplataforma utilizado para comunicação por meio de mensagens, voz e vídeo.

1. Conclusão

O desenvolvimento de projetos é a parte principal da vida profissional de um desenvolvedor, e devido a alta complexidade na produção da documentação e codificação de projetos, a aplicação prática do conteúdo teórico estudado é essencial para garantir que o desenvolvedor terá a experiência necessária para atuar no mercado de trabalho em diversas etapas do projeto.

Desenvolver o aplicativo IBurger para dispositivos móveis foi um processo complexo que buscou seguir com fidelidade os padrões de desenvolvimento de projetos presentes no mercado. Portanto a execução do projeto serviu como fonte de aprendizado prático, pois foram adquiridas experiências que se assemelham com as vivenciados por desenvolvedores profissionais e garantiu aos integrantes do grupo mair compreensão das ferramentas e conceitos utilizados durante o projeto.

1. Bibliografia

**SISTEMA DE BIBLIOTECAS PROF. JOSÉ STORÓPOLI.** Universidade Nove de Julho. Manual para Elaboração de Trabalhos Acadêmicos de acordo com ABNT.

Disponível em :http://docs.uninove.br/arte/pdfs/manual-elaboracao-Trabahos-ABNT.pdf. Acesso em 20 de Setembro de 2020.

**MEDIUM INTEGRAÇÃO DE API COM JAVASCRIPT**

Disponível em : https://medium.com/collabcode/como-criar-um-api-rest-com-javascript-e-nodejs-como-plataforma-1f6c647a5a89

**API.FLUTTER DOCUMENTAÇÃO DE FLUTTER PARA CRIAÇÃO DE APP**

Disponível em : https://api.flutter.dev/

**HEVODATA INTEGRAÇÃO DA API COM SQL SERVER**

Disponível em : https://hevodata.com/learn/sql-server-rest-api-integration/