###### Especificação técnica para demandas Simples

<Número e Nome da Demanda>

<Nome do Sistema>

***Envolvidos***

|  |  |  |
| --- | --- | --- |
| **Usuário(s)/Departamento(s) requisitante(s)** | | **Telefone/Ramal** |
|  | |  |
| **Líder Técnico** | **Líder da Demanda - Fábrica** | |
|  |  | |

# 1. Introdução

A Especificação Técnica descreve a arquitetura técnica e as tecnologias utilizadas no desenvolvimento do aplicativo móvel de gerenciamento de tarefas. Este documento destina-se aos desenvolvedores e engenheiros envolvidos no projeto.

# 2. Arquitetura do Sistema

O aplicativo móvel será desenvolvido seguindo uma arquitetura de cliente-servidor, onde o cliente é o aplicativo instalado nos dispositivos móveis dos usuários e o servidor é responsável pelo armazenamento e gerenciamento dos dados.

## 2.1 Cliente:

O aplicativo cliente será desenvolvido usando a plataforma de desenvolvimento multiplataforma Flutter, que permite a criação de aplicativos nativos para iOS e Android a partir de um único código-base.

O Flutter utiliza a linguagem de programação Dart, que é open source e possui uma comunidade ativa de desenvolvedores.

***Exemplo de código Flutter (Dart):***

*dart*

*Copy code*

*import 'package:flutter/material.dart';*

*void main() {*

*runApp(MyApp());*

*}*

*class MyApp extends StatelessWidget {*

*@override*

*Widget build(BuildContext context) {*

*return MaterialApp(*

*title: 'Gerenciador de Tarefas',*

*theme: ThemeData(*

*primarySwatch: Colors.blue,*

*visualDensity: VisualDensity.adaptivePlatformDensity,*

*),*

*home: MyHomePage(title: 'Tarefas'),*

*);*

*}*

*}*

*class MyHomePage extends StatefulWidget {*

*MyHomePage({Key key, this.title}) : super(key: key);*

*final String title;*

*@override*

*\_MyHomePageState createState() => \_MyHomePageState();*

*}*

*class \_MyHomePageState extends State<MyHomePage> {*

*@override*

*Widget build(BuildContext context) {*

*return Scaffold(*

*appBar: AppBar(*

*title: Text(widget.title),*

*),*

*body: Center(*

*child: Text(*

*'Lista de Tarefas',*

*style: TextStyle(fontSize: 20),*

*),*

*),*

*floatingActionButton: FloatingActionButton(*

*onPressed: () {*

*// Adicionar nova tarefa*

*},*

*tooltip: 'Adicionar Tarefa',*

*child: Icon(Icons.add),*

*),*

*);*

*}*

*}*

## 2.2 Servidor:

O servidor será implementado utilizando a estrutura de desenvolvimento web Django, que é um framework web Python de código aberto, conhecido por sua simplicidade e eficiência.

O Django oferece recursos poderosos para o desenvolvimento de aplicativos web escaláveis e seguros, incluindo um ORM (Object-Relational Mapping) para interagir com o banco de dados e um sistema de autenticação robusto.

***Exemplo de código Django (Python):***

*python*

*Copy code*

*from django.db import models*

*class Task(models.Model):*

*title = models.CharField(max\_length=100)*

*description = models.TextField()*

*deadline = models.DateTimeField()*

*priority = models.IntegerField(default=0)*

*completed = models.BooleanField(default=False)*

*def \_\_str\_\_(self):*

*return self.title*

# 3. Banco de Dados

O banco de dados do aplicativo será desenvolvido usando o sistema de gerenciamento de banco de dados relacional PostgreSQL, que é open source e amplamente utilizado na indústria de desenvolvimento de software.

## 3.1 Estrutura de Dados:

O banco de dados será projetado para armazenar informações sobre usuários, tarefas, notificações e outras entidades relacionadas ao aplicativo.

Exemplo de definição de modelo no Django (Python):

*python*

*Copy code*

*from django.db import models*

*class Task(models.Model):*

*title = models.CharField(max\_length=100)*

*description = models.TextField()*

*deadline = models.DateTimeField()*

*priority = models.IntegerField(default=0)*

*completed = models.BooleanField(default=False)*

*def \_\_str\_\_(self):*

*return self.title*

# 4. Segurança

A segurança do aplicativo será uma prioridade e será implementada utilizando práticas recomendadas de segurança de software, incluindo autenticação de usuário, criptografia de dados e prevenção de ataques comuns, como injeção de SQL e XSS (Cross-Site Scripting).

## 4.1 Autenticação de Usuário:

O Django oferece um sistema de autenticação embutido que pode ser facilmente configurado para proteger as rotas e endpoints do aplicativo.

As senhas dos usuários serão armazenadas de forma segura utilizando funções de hash, como bcrypt.

## 4.2 Criptografia de Dados:

As comunicações entre o aplicativo cliente e o servidor serão protegidas utilizando o protocolo HTTPS para garantir a confidencialidade e integridade dos dados transmitidos.

# 5. Testes e Qualidade de Código

O código do aplicativo será submetido a testes rigorosos usando frameworks de teste automatizado, como o Flutter Test para o cliente e o Django Test para o servidor. Além disso, serão adotadas práticas de desenvolvimento ágil, como revisões de código e integração contínua, para garantir a qualidade e a estabilidade do software.

***Exemplo de teste no Django (Python):***

*python*

*Copy code*

*from django.test import TestCase*

*from .models import Task*

*class TaskModelTest(TestCase):*

*def setUp(self):*

*Task.objects.create(title="Test Task", description="Test Description", deadline="2024-03-20", priority=1, completed=False)*

*def test\_task\_creation(self):*

*task = Task.objects.get(title="Test Task")*

*self.assertEqual(task.description, "Test Description")*

*self.assertEqual(task.priority, 1)*

*self.assertFalse(task.completed)*