

**Documento Técnico da SmartTicket App**

Este documento técnico descreve os requisitos e a arquitetura da Aplicação, uma aplicação móvel projetada para uso em atividades desportivas.

# Requisitos da Aplicação

### Perfil do Funcionário

**O perfil do Funcionário permite o acesso às seguintes funcionalidades:**

* Listar turmas associadas ao professor.
* Visualizar fichas de avaliação dos alunos de cada turma.
* Efetuar avaliações dos alunos pela aplicação.
* Editar e visualizar fichas de avaliação conforme necessário.

### Perfil do Cliente

**O perfil do Cliente oferece as seguintes funcionalidades:**

* Exibir o histórico de avaliações.
* Fazer pedidos de inscrição em aulas.
* Solicitar anulação de inscrição em aulas.
* Visualizar o horário das suas aulas e das aulas disponíveis.
* Ver pagamentos pendentes de quotas.
* Visualizar o histórico dos pagamentos efetuados.
* Selecionar pagamentos pendentes e efetuar pagamentos.
* Pagar quotas de agregados associados ao cliente.
* Solicitar a associação de um novo membro à lista de agregados.

### Funcionalidades Globais da Aplicação

* Envio de push notifications.
* Exibição de informações sobre a entidade a que pertencem funcionários e clientes.
* Visualização e edição do perfil do utilizador.

Esses requisitos estabelecem a base para o desenvolvimento da aplicação e garantem que as necessidades sejam atendidas de maneira eficiente e intuitiva.Framework de Desenvolvimento: Flutter

A aplicação SmartTicketApp será desenvolvida utilizando o framework **Flutter**.

* Flutter é uma estrutura de desenvolvimento de código aberto da Google que permite a criação de aplicativos nativos para dispositivos móveis, web e desktop a partir de um único código-fonte.
* A escolha do Flutter foi feita devido à sua capacidade de fornecer uma experiência de utilizador (UX) consistente e de alta qualidade em várias plataformas.
* O Flutter usa a linguagem de programação **Dart** para a implementação lógica da aplicação.
* Além disso, o Flutter oferece uma ampla variedade de widgets personalizáveis que podem ser usados para criar a interface de utilizador (UI) da SmartTicketApp de forma eficaz e eficiente.

# Plataformas de Disponibilidade: Android e iOS

A SmartTicketApp será desenvolvida para ser disponível nas plataformas Android e iOS.

* O Flutter, como framework de desenvolvimento escolhido, permite a criação de aplicativos nativos para ambas as plataformas a partir de um único código-fonte.
* Isso garante que os utilizadores de dispositivos Android e iOS tenham acesso à aplicação, proporcionando uma ampla base de utilizadores

# Gestão de Estados com RiverPod:

Para a gestão de estados, será utilizado o RiverPod, uma biblioteca de gestão de estados recomendada pelo Flutter.

* O RiverPod simplifica a gestão de estados e a injeção de dependências na aplicação.
* Ele oferece uma solução eficaz para manter e compartilhar estados de forma reativa entre os vários componentes da aplicação, o que é essencial para uma experiência de utilizador fluida e responsiva.
* A escolha do RiverPod contribuirá para um código organizado e fácil de manter, permitindo a expansão e atualização da aplicação de forma eficiente.

# Dependências da Aplicação:

(**Para mais informações sobre os packages utilizados, ir a** <https://pub.dev/>)

A SmartTicketApp utiliza várias bibliotecas e dependências para fornecer funcionalidades específicas e melhorar a experiência do utilizador (UE). Abaixo, listamos as principais dependências utilizadas no desenvolvimento da aplicação:

1. **cupertino\_icons (^1.0.2):** Esta biblioteca fornece ícones específicos para a plataforma iOS, garantindo uma experiência consistente na plataforma.
2. **google\_fonts (^5.1.0):** Utilizada para incorporar fontes personalizadas na aplicação, permitindo um design de texto atraente e consistente.
3. **transparent\_image (^2.0.1):** Esta dependência ajuda no manuseio de imagens transparentes, garantindo uma renderização adequada.
4. **http (^1.1.0):** Usada para realizar solicitações HTTP e interagir com um servidor para buscar e enviar dados.
5. **flutter\_secure\_storage (^8.0.0):** Essa biblioteca fornece armazenamento seguro para dados sensíveis, como tokens de autenticação.
6. **flutter\_riverpod (^2.3.6):** O RiverPod é o gerenciador de estado escolhido para controlar o estado global da aplicação, permitindo uma gestão eficaz dos dados.
7. **connectivity\_plus (^4.0.1):** Utilizada para verificar a conectividade de rede, garantindo que a aplicação funcione corretamente em diferentes cenários de conexão.
8. **intl (qualquer versão):** Essa biblioteca auxilia na internacionalização da aplicação e na formatação de datas e números de acordo com a localização do utilizador.
9. **url\_launcher (^6.1.12):** Utilizada para abrir URLs num navegador externo, permitindo que os utilizadores acessem recursos da web.
10. **badges (^3.1.1):** Essa biblioteca é usada para adicionar distintivos às notificações na barra de navegação, melhorando a experiência do utilizador.
11. **diacritic (^0.1.4):** Ajuda na manipulação de caracteres diacríticos, útil para processamento de texto internacionalizado.
12. **curved\_navigation\_bar (^1.0.3):** Utilizada para criar uma barra de navegação curva e personalizada na aplicação.
13. **flutter\_localizations (sdk: flutter):** SDK do Flutter para suporte a localizações, permitindo a tradução da interface do utilizador.
14. **file\_picker (^5.3.3):** Essa dependência permite que os utilizadores escolham arquivos em seus dispositivos, como parte da funcionalidade da aplicação.
15. **pdf (^3.10.4)** e **open\_file (^3.3.2):** Utilizadas para exibir e abrir arquivos PDF na aplicação, fornecendo recursos de visualização de documentos.
16. **path\_provider (^2.1.0):** Essa biblioteca auxilia na obtenção de diretórios específicos do sistema de arquivos para armazenamento e recuperação de dados.
17. **device\_info\_plus (^9.0.3):** Utilizada para obter informações detalhadas sobre o dispositivo do utilizador, útil para personalização e análise de dispositivos.
18. **flutter\_exit\_app (^1.1.2):** Permite que os utilizadores saiam da aplicação quando necessário.
19. **firebase\_messaging (^14.6.7)** e **firebase\_core (^2.15.1):** Integração com serviços de mensagens Firebase para suporte a notificações e mensagens push.
20. **flutter\_zoom\_drawer (^3.1.1):** Esta biblioteca é usada para criar uma gaveta de navegação personalizada na aplicação.

# Estrutura de Pastas e Ficheiros:

* **lib/main.dart:** O ponto de entrada da aplicação Flutter.
* **lib/models:** Pasta que contém modelos de dados utilizados na aplicação.
  + **lib/models/client:** Modelos de dados relacionados aos clientes.
  + **lib/models/employee:** Modelos de dados relacionados aos funcionários.
  + **lib/models/global:** Modelos de dados globais da aplicação.
* **lib/services:** Pasta que contém classes de serviços, responsáveis pela lógica de negócios e interações com APIs externas.
* **lib/screens:** Pasta que contém as diferentes telas da aplicação.
  + **lib/screens/client:** Telas específicas para o perfil de cliente.
  + **lib/screens/employee:** Telas específicas para o perfil de funcionário.
  + **lib/screens/global:** Telas usadas globalmente na aplicação
* **lib/widgets:** Pasta que contém widgets reutilizáveis usados em toda a aplicação.
* **lib/providers:** Pasta que contém os provedores de estado usados na gestão de estados da aplicação.
  + **lib/providers/client:** Provedores relacionados aos clientes.
  + **lib/providers/employee:** Provedores relacionados aos funcionários.
  + **lib/providers/global:** Provedores globais da aplicação.
* **lib/utils:** Pasta para utilitários e funções auxiliares.
* **lib/constants:** Pasta para constantes, como cores, tamanhos de fonte e strings usadas em todo o aplicativo.

# Autenticação

A aplicação é projetada para autenticar dispositivos e os seus utilizadores, fornecendo acesso a recursos com base nas credenciais e no estado do dispositivo.

## Diagrama de Fluxo de Autenticação

A figura abaixo ilustra o processo de autenticação da aplicação:

Uma imagem com captura de ecrã, texto, design

Descrição gerada automaticamente

**Descrição do Diagrama de Fluxo de Autenticação:**

O processo começa quando o utilizador inicia a aplicação. A primeira verificação é a conectividade à internet. Se o dispositivo estiver online, a aplicação verifica se o ID do dispositivo já está registado. Se sim, o utilizador é levado diretamente para o ecrã inicial.

Se o dispositivo não estiver registado, o utilizador é redirecionado para o ecrã de registo, onde preenche o NIF/Utilizador e o endereço de e-mail. A aplicação envia um pedido ao servidor com esses dados e recebe um link de serviço da web como resposta. Esse link é armazenado usando o Secure Storage fornecido pelo Flutter.

Em seguida, a aplicação verifica novamente o estado do dispositivo. Se o dispositivo estiver ativado, o utilizador é direcionado para o ecrã inicial. Caso contrário, é exibido um campo de ativação, onde o utilizador insere um código recebido por e-mail. Se o código estiver correto, o dispositivo é registado e ativado, e o utilizador é levado para o ecrã inicial. Se o código estiver incorreto, é exibida uma mensagem de erro.

Isso garante que apenas dispositivos e utilizadores autorizados tenham acesso aos recursos da SmartTicketApp, protegendo a segurança e a integridade dos dados da aplicação.