Tecnologias Utilizadas:

Flutter
Framework de desenvolvimento multiplataforma que permite criar aplicações nativas para Android e iOS utilizando uma única base de código.

Versão: 3.24.5

ValueNotifier
Utilizado como gerenciador de estados para manter a simplicidade e o desempenho do aplicativo, facilitando a atualização da interface do usuário em tempo real.

Firebase Firestore
Banco de dados NoSQL baseado na nuvem, utilizado para armazenar e sincronizar os dados do aplicativo.

Instruções para acessar o sistema:

Pré-requisitos
Dispositivo móvel com sistema operacional Android.
Conexão ativa com a internet.

Instalação do APK
Navegue até a pasta apk-build disponível no repositório do projeto.
Transfira o arquivo APK para o dispositivo móvel.
No dispositivo, permita a instalação de aplicativos de fontes desconhecidas (se necessário).
Execute o APK para instalar o aplicativo.

Execução
Certifique-se de estar conectado à internet antes de abrir o aplicativo.
O aplicativo utilizará a conexão para acessar os dados no Firebase Firestore.

Observações:

Foi utilizado um banco de dados NoSQL para evitar a necessidade de desenvolver uma API, o que demandaria mais tempo.

Devido à minha pouca experiência com a tecnologia (Firebase), algumas funcionalidades ficaram de fora neste momento.

O sistema está aberto a otimizações, como uma melhor gestão de interface, implementação de mensagens de loading, tratamento de erros, registros de logs, gerenciamento de agendamentos por usuário e diretivas de acesso. Além disso, o back-end pode ser substituído futuramente por uma solução mais escalável, abrangente e segura.

Tabela de requisições feitas no Sistema com SQL.

Como a tecnologia de back-end escolhida foi uma base NoSQL, preparei algumas comparações de consultas (queries) que seriam realizadas caso o sistema utilizasse um back-end SQL, considerando minha maior facilidade e experiência com essa tecnologia.

Algumas dessas consultas não estão implementadas no sistema atual, mas servem como ideias para futuras implementações.

Tabelas do banco de dados:

```
CREATE TABLE Users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    contactNumber VARCHAR(50),
    username VARCHAR(100) UNIQUE NOT NULL,
    password VARCHAR(255) NOT NULL,
    userType VARCHAR(50) NOT NULL
);

CREATE TABLE Scheduling (
    id INT AUTO_INCREMENT PRIMARY KEY,
    idUser INT NOT NULL,
    serviceType VARCHAR(255) NOT NULL,
```

```
    dateHour DATETIME NOT NULL,
    status VARCHAR(50) NOT NULL,
    FOREIGN KEY (idUser) REFERENCES Users(id) ON DELETE CASCADE
);
```

Sistema	**SQL**
pega todos os registros de Agendamentos	SELECT * FROM Scheduling;
pega Agendamento por Status	SELECT u.name AS UserName, u.contactNumber AS UserContact, s.serviceType AS Service, s.dateHour AS DateHour, s.status AS Status FROM Scheduling s JOIN Users u ON s.idUser = u.id WHERE s.status = ?;
pega Agendamentos que estão entre as datas desejadas	SELECT u.name AS UserName, u.contactNumber AS UserContact, s.serviceType AS Service, s.dateHour AS DateHour, s.status AS Status FROM Scheduling s JOIN Users u ON s.idUser = u.id

	WHERE s.dateHour BETWEEN ? AND ?;
pega Usuário por Id	SELECT name, contactNumber, username, usertype FROM Users WHERE id = ?;
pega Agendamento que está na mesma semana que a data desejada	SELECT u.name AS UserName, u.contactNumber AS UserContact, s.serviceType AS Service, s.dateHour AS DateHour, s.status AS Status FROM Scheduling s JOIN Users u ON s.idUser = u.id WHERE YEAR(s.dateHour) = YEAR(?) AND WEEK(s.dateHour, 1) = WEEK(?, 1);
pega Agendamento por Id	SELECT u.name AS UserName, u.contactNumber AS UserContact, s.serviceType AS Service, s.dateHour AS DateHour, s.status AS Status

	FROM Scheduling s JOIN Users u ON s.idUser = u.id WHERE s.id = ?;
pega Agendamento por Usuário	SELECT u.name AS UserName, u.contactNumber AS UserContact, s.serviceType AS Service, s.dateHour AS DateHour, s.status AS Status FROM Scheduling s JOIN Users u ON s.idUser = u.id WHERE u.id = ?;
verifica quantidade de Agendamentos feitos em uma semana específica	SELECT COUNT(*) AS TotalAppointments FROM Scheduling s WHERE YEAR(s.dateHour) = YEAR(?) AND WEEK(s.dateHour, 1) = WEEK(?, 1);
verifica quantidade de agendamentos confirmados entre uma data específica	SELECT COUNT(*) AS TotalAppointments FROM Scheduling s WHERE s.status = 'Confirmado' AND s.dateHour BETWEEN ? AND ?;
verifica quantidade de Agendamentos com o tipo de serviço Corte feitos entre uma data específica	SELECT COUNT(*) AS TotalAppointments FROM Scheduling s WHERE s.serviceType = 'Corte' AND s.dateHour BETWEEN ? AND ?;