# Manual Code
# Amigos dos Jardinetes

### 1.0: Introdução:

O documento a seguir se refere a informações relevantes sobre o código, inclusive o próprio programa, Amigos dos Jardinetes. Use-o como guia quando for necessário, e sempre busque mantê-lo atualizado. Em meados de Setembro/2025, o Jardinetes v2.0 deve ser lançado e com ele uma nova documentação. Considerando que parte significativa dela será baseada na versão legado do Amigos do Jardinetes, essa documentação permanecerá importante na manutenção e atualização do código.

### 1.0.1: Outras fontes de informação:

Caso precise de mais informações ou referências para a estruturação do código e futuras alterações, seguem algumas referências:

- RoadMap SQL[1]: https://roadmap.sh/
- Tech Guide da Alura[2]: https://techguide.sh/

### 1.1: Sobre o versionamento:

Parte importante desse manual, é definir algumas diretrizes para a atualização do código, começando pelas regras de versionamento, que no momento não existem.

Ao commitar o código no GitHub, ou qualquer outro repositório, sua nomenclatura deverá iniciar com o nome "**amigos_jardinetes_v**", seguido então da versão atual do programa, essa versão deverá ser atualizada apenas quando grandes mudanças na estrutura do código forem realizadas.

A versão do programa, será seguida por um ponto, e então outro número, referente ao mês corrente que aquele código foi commitado, sendo que o mês 0 é definido sempre que uma nova versão, ou seja, uma grande atualização, é lançada. Segue a seguir os meses que as versões do Jardinetes foram lançadas:

- **Amigos dos Jardinetes v1.0.0:** Janeiro 2024.
- **Amigos dos Jardinetes v2.0.0:** Em breve...

Após a identificação da versão e do mês atual do commit, o nome deve possuir outro ponto e então ser identificado pela quantidade de commits realizados naquele mês, de forma que se inicie em 0 no primeiro commit do mês, e então, sempre que algo for alterado no código do repositório se adicione +1 a esse valor.

Por fim, caso a alteração seja pequena ou de pouca importância e tenha relação com o último commit, um novo ponto pode ser adicionado ao final e após ele uma identificação da alteração feita, podendo essa ser alguma palavra, número, sigla, etc.

Segue abaixos alguns exemplos de nomenclatura aceitável no código:

- **amigos_jardinetes_v1.0.0** (primeira versão do código)
- **amigos_jardinetes_v1.5.0** (primeira atualização no mês 5 após o lançamento da 1ª versão)

- **amigos_jardinetes_v1.5.6** (sexta atualização do mesmo mês 5 acima)
- **amigos_jardinetes_v1.5.6.fonte** (pequena alteração na atualização anterior)
- **amigos_jardinetes_v2.1.0** (primeira atualização no mês 1 após o lançamento da 2ª versão)

### 1.2: Sobre a documentação:

A documentação será composta por esse arquivo de texto, o manual de identidade visual e um diagrama de classes referente ao código.

A nomenclatura deles será feita seguindo os prefixos "**README_jardinetes**". "**IDV_jardinetes**" e "**classes_jardinetes**", seguida então do mês e ano da nova versão. A exceção ocorre para as "**classes_jardinetes**", que no lugar do mês e ano da versão, será seguido pela versão do código no qual ele foi baseado, conforme as regras descritas na seção 1.1 deste documento.

Esses arquivos deverão ser enviados para o drive do Jardinetes, em suas respectivas pastas, com as versões mais recentes desse texto e do diagrama de classes também devendo ser incluídos na pasta "Documentação" do código no GitHub.

Outros arquivos e materiais também poderão ser acessados no drive do Jardinetes, sendo que esse deve ser mantido atualizado.

## 2.0: Requisitos necessários para rodar o projeto:

As ferramentas utilizadas para manutenção, compilação e exportação do código, são compostas por:

- GitHub
- VS Code
- Docker

### 2.1: Configurações do GitHub:

Segue abaixo o passo a passo para a configuração do GitHub no seu ambiente de trabalho:

- Crie uma conta no GitHub.
- Conceda acesso ao repositório: usando a conta do Amigos dos Jardinetes, atribua o novo Contribuidor no repositório Jardinetes. Para isso, abra o repositório e vá para Settings/Collaborators/Add people.
- Instale o VS Code na sua máquina.
- Instale o GitHub na sua máquina e faça login.
- Clone o repositório para a sua máquina.
- Baixe as atualizações do repositório através do "Fetch origin".
- Abra o projeto através do editor externo, se seguir esse texto será o VS Code.

### 2.2: Configurações e extensões do VS Code:

Após seguir os passos anteriores, agora será a vez de configurar o seu espaço de trabalho no editor, para isso, será preciso instalar algumas coisas, siga as instruções abaixo:

- Instale o Windows PowerShell na sua máquina.
- Instale o GitBash na sua máquina.
- Com o terminal PowerShell aberto, para configurar a sua máquina, rode:

```
None
Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope CurrentUser
```

- E então verifique a versão do Node.js, npm, expo e yarn com:

```
None
node -v
npm -v
expo --version
yarn --version
```

- Caso o Node.js e o npm não estejam instalados, baixar a versão LTS (é mais estável) em https://nodejs.org, e então verifique a instalação no PowerShell com:

```
None
node -v
npm -v
```

- Caso expo e yarn não estejam instalados ou estejam desatualizados seguir os passos a seguir:

```
None
npm install -g yarn
yarn -v

cd Jardinetes
npm install
npx expo install
```

- Por fim, só rodar a versão de teste local no navegador:

```
None
npx expo start --web
```

*os procedimentos do VS Code variam bastante de máquina para máquina, então dependendo o caso essas instruções podem não funcionar como o esperado. Se isso acontecer, peça ajuda e pesquise sobre, existem vários fóruns que tratam do assunto, e IAs também costumam ser competentes nesse assunto.

### 2.3: Configurações do Docker:

...

### 2.4: Outras ferramentas:

Para a criação do Diagrama de Classes foram usados o Visual Paradigm Online, visto que ele é gratuito , mas se preferir pode usar outro, desde que siga os procedimentos estabelecidos.

Para o Design, foram utilizados o Photoshop, Canva e o Figma, mas assim como o caso acima, outras ferramentas também podem ser utilizadas.

## 3.0: Informações sobre o projeto:

Esse tópico consiste em informações sobre o projeto, ou código, do Jardinetes, e informações úteis para se trabalhar com ele.

### 3.1: Como se orientar dentro do projeto:

### 3.2: Estrutura do código:

Segue a estrutura do código, nomes que começam com "./" se referem a pastas dentro do código, enquanto os demais se referem a arquivos.

**./.expo**

Possui configurações internas e cache para utilização da ferramenta e plataforma Expo.

**./.history**

Criada por extensões do VS Code, serve como backup local, mantendo um histórico automático de versões dos arquivos. Pelo GitHub já servir como backup e possuir as versões antigas do código, colocar o .history no .gitignore para não versionar isso no repositório.

**./.vscode**

Possui configurações específicas do VS Code, ajudando a manter as configurações de execução compartilhadas entre a equipe.

**./android**

Essa pasta é criada automaticamente quando você tem um projeto React Native (ou quando usa Expo e depois faz o *eject* para acesso nativo).

Ela contém todo o código nativo Android necessário para o app rodar no sistema.

Principais arquivos:

- **./app/** → código da aplicação Android em Java/Kotlin.
- **./gradle/** → arquivos de configuração do Gradle (sistema de build do Android).

- **.gitignore** → lista de arquivos/pastas a serem ignorados no Git (no escopo do Android).
- **build.gradle** → script principal do Gradle, define dependências e configurações do projeto Android.
- **gradle.properties** → variáveis de configuração (como versões de SDK, flags de desempenho etc.).
- **gradlew** → script para rodar o Gradle no Linux/macOS.
- **gradlew.bat** → script para rodar o Gradle no Windows.
- **settings.gradle** → indica quais módulos fazem parte do build (normalmente o app).

**./assets**

Essa pasta guarda recursos estáticos do app (imagens, ícones, splash screen etc.).

**./dist**

A pasta dist (de distribution) é criada quando você faz o build de produção. Aqui ficam os arquivos finais que serão enviados para o servidor ou usados no app.

Subpastas:

- **asset-manifest.json** → lista os arquivos gerados no build.
- **index.html** → página principal do app quando rodado como web.
- **manifest.json** → usado em PWA (Progressive Web App), define nome, ícones, tema etc.
- **serve.json** → configuração do servidor local que vai servir os arquivos.

**./env**

Essa pasta contém arquivos de configuração de ambiente (como URLs de API, chaves secretas, configurações específicas para dev/test/prod).

**./node_modules**

Onde ficam todas as dependências do seu projeto instaladas via npm ou yarn. Ou seja, é onde ficam as bibliotecase e dependências internas delas que foram usadas durante o projeto. Você nunca edita nada dentro dessa pasta, mas caso o faça ou apague ela, basta rodar npm install e reinstalá-la.

Por ela ser gigante, e conter dezenas de MB, ela não deve ir para o Git, sendo inclusa no .gitignore.

**./public**

Aqui ficam os arquivos públicos que podem ser acessados diretamente pelo navegador.

**./scripts**

Pasta com arquivos e configurações relacionadas aos containers e ao deploy no sistema da UTFPR. Possui outros scripts auxiliares.

**./src**

Essa é a pasta mais importante: guarda o código-fonte real da aplicação.

Dentro dela:

- **./assets/** → recursos usados dentro do código (imagens, ícones etc. que não são públicos).
- **./authContext/** → contém Context API do React para gerenciar autenticação (login/logout, usuário atual, permissões).

- **./data/** → dados estáticos, JSONs ou utilitários para acesso a dados.
- **./firebase/** → configurações e serviços do Firebase (ex.: autenticação, banco Firestore, storage).
- **./pages/** → componentes/páginas principais do app (ex.: Login, Dashboard, Home etc.).
- **./routes/** → define o roteamento da aplicação (usando react-router-dom).

**.env:** Variáveis de ambiente.
**.firebaserc:** Seleção de projeto Firebase.
**.gitattributes:** Usado para definir atributos e regras especiais para determinados arquivos no repositório, controlando como os arquivos do repositório se comportam.
**.gitignore:** Usado para ignorar arquivos e pastas durante o versionamento. Impede que determinados arquivos subam para o repositório do Git. Sempre verificar antes de realizar o commit.
**App.js:** Ponto de entrada do app React/React Native.
**app.json:** Arquivo para configuração do Expo.
**babel.config.js:** Configuração do Babel.
**compose.yaml:** Orquestração com outros serviços.
**deploy.sh:** Script automatizado para deploy.
**Dockerfile:** README do Docker.
**eas.json:** Configuração de build Expo (EAS).
**firebase-debug-log:** Log do firebase.
**firebase.json:** Configura hosting/functions.
**firestore.indexes.json lock:** Índices do banco de dados.
**firestore.rules:** Regras de segurança.
**function.js:** Define algumas funções do código.
**google-services.json:** Configuração Android.
**package-lock.json:** Define dependências e scripts.
**package.json:** Define dependências e scripts.
**vercel.json:** Configuração do deploy na Vercel.
**web-build.zip:** Pacote de build final para web.
**yarn.lock:** Define dependências e scripts.

### 4.0: Demais arquivos dentro do código:

Esses são a raiz do projeto, com os arquivos mais importantes. Iremos dissecá-los a seguir.