■ DEPLOY CHEFCODE 2.0 — Guida completa

Questa guida è un promemoria condivisibile per il team: descrive il flusso di sviluppo locale, il ve
rsionamento su GitHub, il deploy su Render e la generazione dell'APK Android. Contiene comandi, rif
rimenti ai file del progetto e verifiche pratiche.

Indice

- Panoramica progetto
- Prerequisiti
- Sviluppo locale (quick start)
- GitHub: branch, commit e push
- Deploy su Render (configurazione e verifica)
- Build APK (Debug & Release)
- Checklist di sicurezza
- FAQ e risorse

Panoramica progetto

Cartelle principali:
- `backend/` — server Node.js (entry: `backend/server.js`)
- `frontend/mobile/` — app mobile con directory `android/`
- `www/` — frontend statico
- Script utili: `BUILD_APK.bat`, `START_CHEFCODE.bat`

Prerequisiti

Ambiente di sviluppo (Windows):
- Git
- Node.js (>= 14)
- npm / yarn
- Java JDK 17 (es. `C:\Program Files\Microsoft\jdk-17.0.16.8-hotspot`)
- Android SDK (impostare `ANDROID_HOME`)
- Android Studio (consigliato)
- ADB (in `platform-tools`)

Account necessari:
- GitHub (accesso repository `chefcode-backend`)
- Render (per deploy)
- Google Play Console (se si pubblica su Play Store)

Sviluppo locale (quick start)

1) Apri il progetto:

```
cd "c:\Users\ambru\Desktop\CHEFCODE\chefcode APP\ChefCode"
```

2) Backend (esempio):

```
cd backend
npm install
# crea .env o imposta variabili per la sessione
set OPENAI_API_KEY="tua_openai_key"
node server.js
```

Nota: `set` imposta la variabile solo per la sessione corrente; `setx` la rende persistente per l
'utente.

3) Frontend statico per test rapido:

```
cd www
python -m http.server 8080
```

4) Verifiche rapide:
- Health endpoint backend: http://localhost:<porta>/health (controlla `backend/server.js`)
- Endpoint AI: `/api/chatgpt-smart`

GitHub: branch, commit e push

Buone pratiche:
- Non inserire API key nel repo (.env è in `.gitignore`).

- Lavora su branch feature e usa Pull Request per le revisioni.

Comandi base:

```
cd "c:\Users\ambru\Desktop\CHEFCODE\chefcode APP\ChefCode"
git checkout -b feature/nome-feature
git add .
git commit -m "Descrizione delle modifiche"
git push origin feature/nome-feature
```

Se il repository è configurato con auto-deploy su `master`, il merge su `master` avvierà il deploy s u Render.

Deploy su Render

1) Verifica su Render Dashboard che il servizio sia collegato al repo.
2) Imposta le environment variables (Settings > Environment):

- OPENAI_API_KEY = <la tua chiave OpenAI>
- Eventuali altre variabili necessarie (controlla `backend/server.js` e `config.json`)

3) Modalità di deploy:
- Auto-deploy: push/merge su `master` → build & deploy automatico.
- Manual: trigger dal dashboard Render.

4) URL di produzione (esempio):

```
https://chefcode-backend-1.onrender.com
```

5) Controlli post-deploy:
- Controlla l'health endpoint in produzione
- Controlla i logs su Render (Dashboard > Logs)

Build APK (Windows) — `BUILD_APK.bat`

Nel repository è incluso `BUILD_APK.bat`, che automatizza i passi seguenti. Lo script:
- imposta `JAVA_HOME` e `ANDROID_HOME`
- esegue `gradlew clean` e `gradlew assembleDebug` o `assembleRelease`

Procedura:

1) Apri PowerShell nella cartella del progetto:

```
cd "c:\Users\ambru\Desktop\CHEFCODE\chefcode APP\ChefCode"
.\BUILD_APK.bat
```

2) Seleziona Debug o Release quando richiesto.

3) Output:
- Debug APK: `frontend\mobile\android\app\build\outputs\apk\debug\app-debug.apk`
- Release APK: `frontend\mobile\android\app\build\outputs\apk\release\app-release.apk`

4) Installazione su device:

```
adb install "<path-to-apk>"
```

5) Note per Release:
- Configura il keystore (in `frontend/mobile/android/app/build.gradle` e/o `gradle.properties`)
- Firmare l'APK con il keystore prima della pubblicazione

Checklist di sicurezza prima del deploy

- Verificare che `.env` e file sensibili non siano nel repo
- Assicurarsi che tutte le API key siano impostate nelle env di Render
- Eseguire test funzionali principali in staging/produzione
- Controllare logs per errori o leak

FAQ rapide

Q: Dove imposto la chiave OpenAI?
A: Su Render: Dashboard > Service > Settings > Environment > Add Variable: `OPENAI_API_KEY`.

Q: Il deploy non parte dopo il push?
A: Controlla che il webhook GitHub sia attivo e che Render sia collegato al repo; verifica i logs de
l deploy.

Risorse

- Render docs: https://render.com/docs
- Android build: https://developer.android.com/studio/build
- GitHub docs: https://docs.github.com

Se vuoi, posso generare una versione PDF o aggiungere screenshot e snippet di `backend/server.js`
r mostrare le env richieste.

■ Guida completa: dallo sviluppo al deploy e build APK

Questa guida è pensata come promemoria condivisibile per il team: copre il flusso di sviluppo (local
e), il versionamento su GitHub, il deploy su Render e la generazione dell'APK per Android. È scritta
in italiano e contiene comandi e riferimenti ai file/directory del progetto.

Indice
- Panoramica progetto
- Prerequisiti (strumenti e accessi)
- Workflow di sviluppo locale
- Versionamento su GitHub (push + pull requests)
- Configurazione e deploy su Render
- Build dell'APK (debug e release)
- Controlli di sicurezza e checklist prima del deploy
- FAQ rapide e risorse utili

Panoramica progetto

Struttura principale (rilevante):
- `backend/` – server Node.js (file principale: `backend/server.js`)
- `frontend/mobile/` – progetto mobile (Android) con directory `android/`
- `www/` – frontend statico usato per demo/hosting
- script utili: `BUILD_APK.bat`, `START_CHEFCODE.bat`, `BUILD_APK.bat`

Prerequisiti

Strumenti locali (Windows):
- Git (configurato con le tue credenziali)
- Node.js (versione consigliata >= 14)
- npm / yarn
- Java JDK 17 (per build Android) – la macchina di riferimento usa: `C:\Program Files\Microsoft\jdk-17.0.16.8-hotspot`
- Android SDK (impostare `ANDROID_HOME`)
- Android Studio (opzionale ma consigliato per debugging/keystore)
- ADB (in `platform-tools`) per installare APK su device

Accessi e account:
- Account GitHub con accesso al repository `chefcode-backend`
- Account Render (https://render.com) con servizio `chefcode-backend` collegato al repo
- (Per pubblicare su Play Store) account Google Play Console

Workflow di sviluppo locale

1. Clona il repository (se non l'hai già):

```
cd "c:\Users\ambru\Desktop\CHEFCODE\chefcode APP\ChefCode"
git clone <repo-url> .
```

2. Esegui il backend localmente (per testare API):

```
cd backend
npm install
copy con .env  (Windows: creare .env con variabili, o usare un editor)
setx OPENAI_API_KEY "tua_openai_key"
node server.js
```

Nota: in sviluppo puoi esportare la variabile solo per la sessione con `set` invece che `setx`.

3. Servire il frontend statico per debug rapido:

```
cd www
python -m http.server 8080
```

4. Verifiche rapide:
- Backend health: http://localhost:.../health (vedi `backend/server.js` per endpoint esatto)
- Endpoint AI: `/api/chatgpt-smart`

Versionamento su GitHub

Regole pratiche:
- Non commitare API keys o file sensibili. `.gitignore` contiene `.env`.
- Usa branch feature/topic e apri Pull Request per merge su `master`.

Comandi base (nella root del progetto):

```
cd "c:\Users\ambru\Desktop\CHEFCODE\chefcode APP\ChefCode"
git add .
git commit -m "Descrizione delle modifiche"
git push origin <nome-branch>
```

Se il deploy automatico è attivo su Render, il push su `master` scatenerà il deploy.

Configurazione e deploy su Render

1. Verifica che il servizio backend sia collegato al repo su Render (Dashboard > Services).
2. Configura le Environment Variables su Render:

- OPENAI_API_KEY = <la tua chiave OpenAI>
- Eventuali altre variabili richieste (controlla `backend/server.js` e `config.json`)

3. Strategie di deploy:
- Auto-deploy (consigliato): ogni push su `master` costruisce e deploya automaticamente.
- Manual deploy: dal dashboard Render puoi triggerare un nuovo deploy.

4. URL di produzione (esempio dal progetto):

```
https://chefcode-backend-1.onrender.com
```

5. Controlli post-deploy:
- Visita l'URL di health/endpoint per verificare
- Controlla i logs su Render (Dashboard > Logs) per errori

Build dell'APK (Windows) – Uso di `BUILD_APK.bat`

Nel progetto è incluso uno script automatico `BUILD_APK.bat` che imposta variabili e chiama Gradle.

1. Prerequisiti macchina di build:
- Java JDK 17
- Android SDK (ANDOID_HOME impostato)
- Gradle wrapper incluso nel progetto Android (`frontend/mobile/android/gradlew`)

2. Come eseguire (apri PowerShell come amministratore se necessario):

```
cd "c:\Users\ambru\Desktop\CHEFCODE\chefcode APP\ChefCode"
.\BUILD_APK.bat
```

Lo script chiederà se vuoi build Debug o Release e poi eseguirà `gradlew assembleDebug` o `gradlew assembleRelease`.

3. Output e posizioni:
- Debug APK: `frontend\mobile\android\app\build\outputs\apk\debug\app-debug.apk`
- Release APK: `frontend\mobile\android\app\build\outputs\apk\release\app-release.apk`

4. Installazione su device collegato:

```
adb install "path\to\app-debug.apk"
```

5. Notes per build release:

- Per una release reale, configura il keystore in `frontend/mobile/android/app/build.gradle` e le pr oprietà in `gradle.properties` o `local.properties` secondo le tue policy.

Controlli di sicurezza e checklist prima del deploy

- Verificare che `.env` non sia nel repo e che le API keys siano impostate solo su Render
- Rivedere i log dopo deploy per eventuali errori o leak
- Eseguire test funzionali principali (login, chiamate AI, salvataggio dati)
- Per la pubblicazione su Play Store, controllare dimensione APK e policy privacy

FAQ rapide

Q: Dove imposto la chiave OpenAI?
A: Su Render: Dashboard > Service > Settings > Environment > Add Variable: `OPENAI_API_KEY`.

Q: Il deploy non parte dopo il push?
A: Controlla che il webhook GitHub sia attivo e che Render sia collegato al repo; controlla i logs d el deploy sul dashboard.

Risorse utili

- Render docs: https://render.com/docs
- Android build: https://developer.android.com/studio/build
- GitHub: https://docs.github.com

Se vuoi, posso:
- aggiungere screenshot/immagini per rendere la guida più visiva;
- trasformarla in un PDF pronto per la condivisione;
- estrarre e includere esatti snippet di `backend/server.js` o `config.json` che mostrano le environ ment variable richieste.

Aggiorno ora la checklist TODO e faccio una verifica rapida del file scritto.