Teknisk Dokumentation
Team04 Tävlingssystem

Team04

16 februari 2026

Innehåll

1 Introduktion

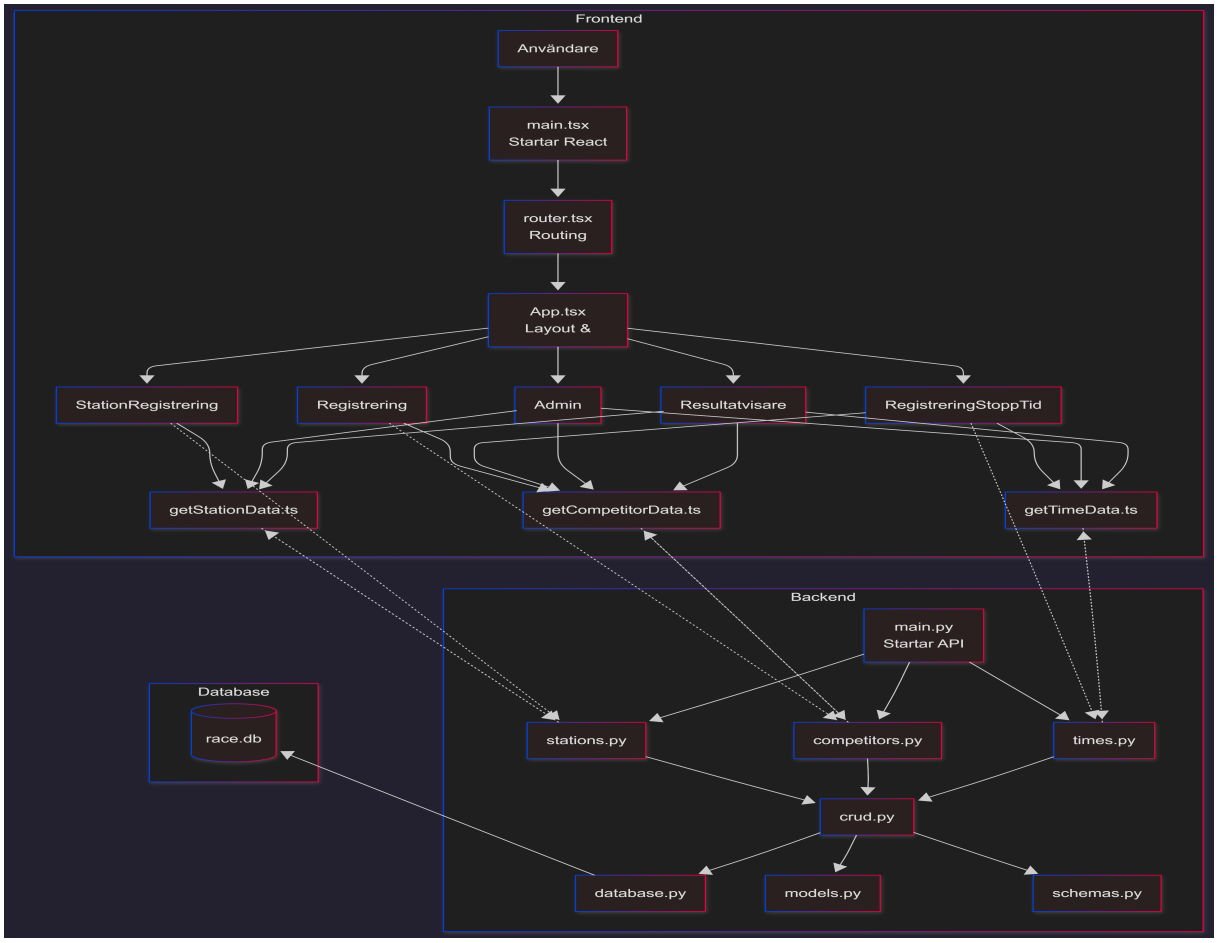
Detta dokument beskriver den överordnade arkitekturen för ett tävlingshanteringssystem. Syftet är att ge en teknisk överblick för framtida utvecklingsteam som ska vidareutveckla eller underhålla systemet.

Dokumentet beskriver:

- Systemets övergripande arkitektur
- Ingående program och filer
- Bygg- och körinstruktioner
- Intern struktur och ansvarsfördelning
- Använda designprinciper och mönster

2 Övergripande Arkitektur och filer

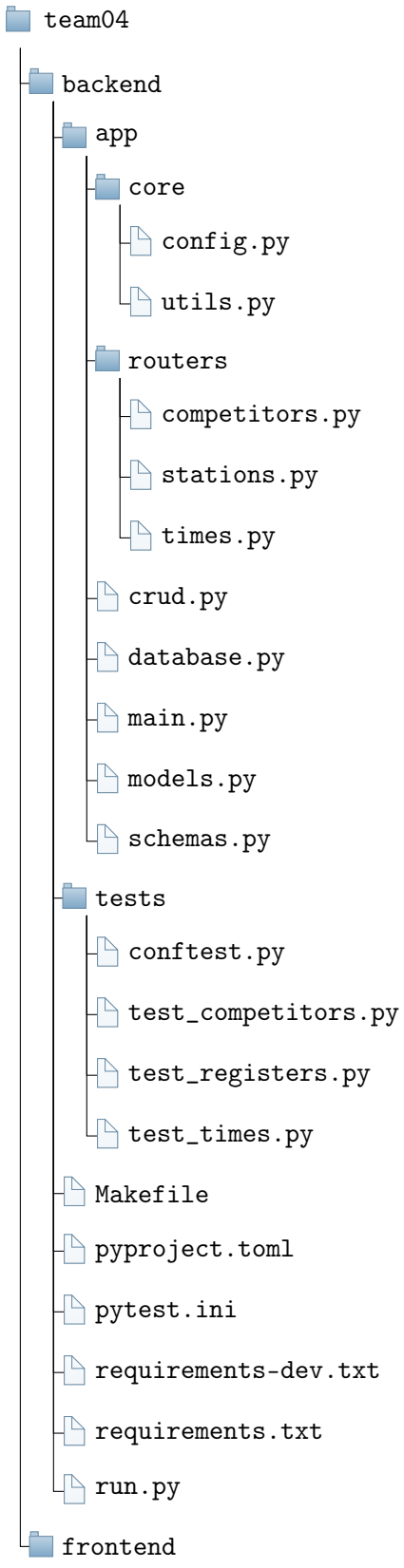
Systemet är uppbyggt enligt en klient–server-arkitektur och består av två huvuddelar: Backend och frontend.

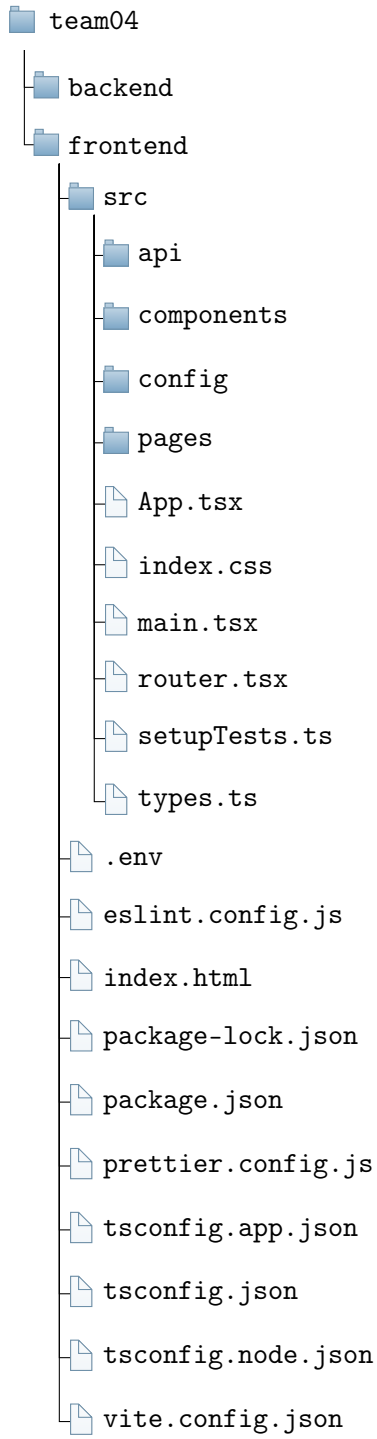


Figur 1: UML av systemets arkitektur. Frontend ansvarar för presentation och användarinteraktion. Backend ansvarar för logik, datavalidering och datalagring.

2.1 Systemets Program och Filer

Nedan visas huvudrepots struktur:




team04
backend
frontend
src
api
components
config
pages
App.tsx
index.css
main.tsx
router.tsx
setupTests.ts
types.ts
.env
eslint.config.js
index.html
package-lock.json
package.json
prettier.config.js
tsconfig.app.json
tsconfig.json
tsconfig.node.json
vite.config.json

2.1.1 Backend

Backend är implementerad i Python med FastAPI och SQLAlchemy.

Fil/Mapp	Ansvar
app/	Applikationskod
models.py	Databasmodeller
schemas.py	Pydantic-scheman (API-kontrakt)
crud.py	Databasoperationer
database.py	Databasanslutning
routers/	API-endpoints
run.py	Startar backend-servern
tests/	Backend-tester

2.1.2 Frontend

Frontend är implementerad i React med TypeScript och Vite.

Fil/Mapp	Ansvar
src/pages/	Applikationens sidor
src/components/	Återanvändbara UI-komponenter
src/api/	API-anrop till backend
router.tsx	Routing-konfiguration
package.json	Projektkonfiguration

3 Bygg och Körning

3.1 Backend

```
cd backend
make install
make run
```

Alternativt:

```
pip install -r requirements.txt
python run.py
```

Vänligen notera att det finns en ytterligare fil `requirements-dev.txt` som innehåller utvecklingsberoenden, inklusive testverktyg. Dessa är inte nödvändiga, men används av teamet och är ett smidigt verktyg. För att installera dessa kan följande kommando användas innan du kör python-filen:

```
pip install -r requirements-dev.txt
```

3.2 Frontend

```
cd frontend
npm install
npm run dev
```

4 Backendens Interna Struktur

Backend är uppdelad enligt ett lagerbaserat mönster:

```
API (Routers)
      |
      v
CRUD-lager
      |
      v
Databas (SQLAlchemy)
```

4.1 Ansvarsfördelning

- **Routers**: Hanterar HTTP-anrop och returnerar JSON-respons.
- **Schemas**: Definierar datamodeller för API-kommunikation.
- **CRUD**: Utför databasoperationer.
- **Models**: Definierar databasens struktur.

5 Frontendens Interna Struktur

Frontend är uppbyggd enligt en komponentbaserad arkitektur.

```
Pages
  |
  v
Components
  |
  v
API-layer
```

5.1 Ansvarsfördelning

- **Pages**: Representerar applikationens olika vyer.
- **Components**: Återanvändbara UI-komponenter.
- **API**: Abstraktion för kommunikation med backend.

6 Kommunikation mellan Delarna

Kommunikationen sker via REST API över HTTP.
Flöde:

1. Frontend skickar HTTP-request.
2. Router tar emot anropet.
3. CRUD-lagret utför databasoperation.
4. Resultat returneras som JSON. **ADD STRUCTURE OF JSON HERE**
5. Frontend uppdaterar användargränssnittet.

---

7 Distribution under Tävling

Under en tävling distribueras systemet enligt följande:

- Backend körs på en central server.
- Frontend körs i webbläsare på klientdatorer.
- Kommunikation sker via lokalt nätverk.

---

8 Vidareutveckling

För att lägga till ny funktionalitet eller uppdatera denna dokumentationen kan följande sektioner hjälpleda utvecklingen.

Vänligen notera att det praktiseras **Test Driven Development** och förutsätts att man följer det för de delar listad nedan. Detta medför även att refaktorisering ska göras **CONTINUE HERE**

8.1 Backend

1. Skapa ny modell i `models.py` vid behov.
2. Skapa motsvarande schema i `schemas.py`.
3. Implementera CRUD-funktion.
4. Lägg till endpoint i `routers/`.

8.2 Frontend

1. Skapa ny sida i `pages/`.
2. Implementera nödvändiga komponenter.
3. Lägg till API-anrop i `api/`.
4. Registrera route i `router.tsx`.

8.3 Teknisk Dokumentation

1. Uppdatera arkitekturdiagram vid större förändringar.
2. Lägg till beskrivningar av nya moduler och filer.
3. Dokumentera API-kontrakt i `schemas.py`.

---

9 Sammanfattning

Systemet är modulärt uppbyggt med tydlig separation mellan presentation, logik och datalager.

Detta dokument syftar till att ge framtida utvecklingsteam en snabb och strukturerad introduktion till systemets uppbyggnad.