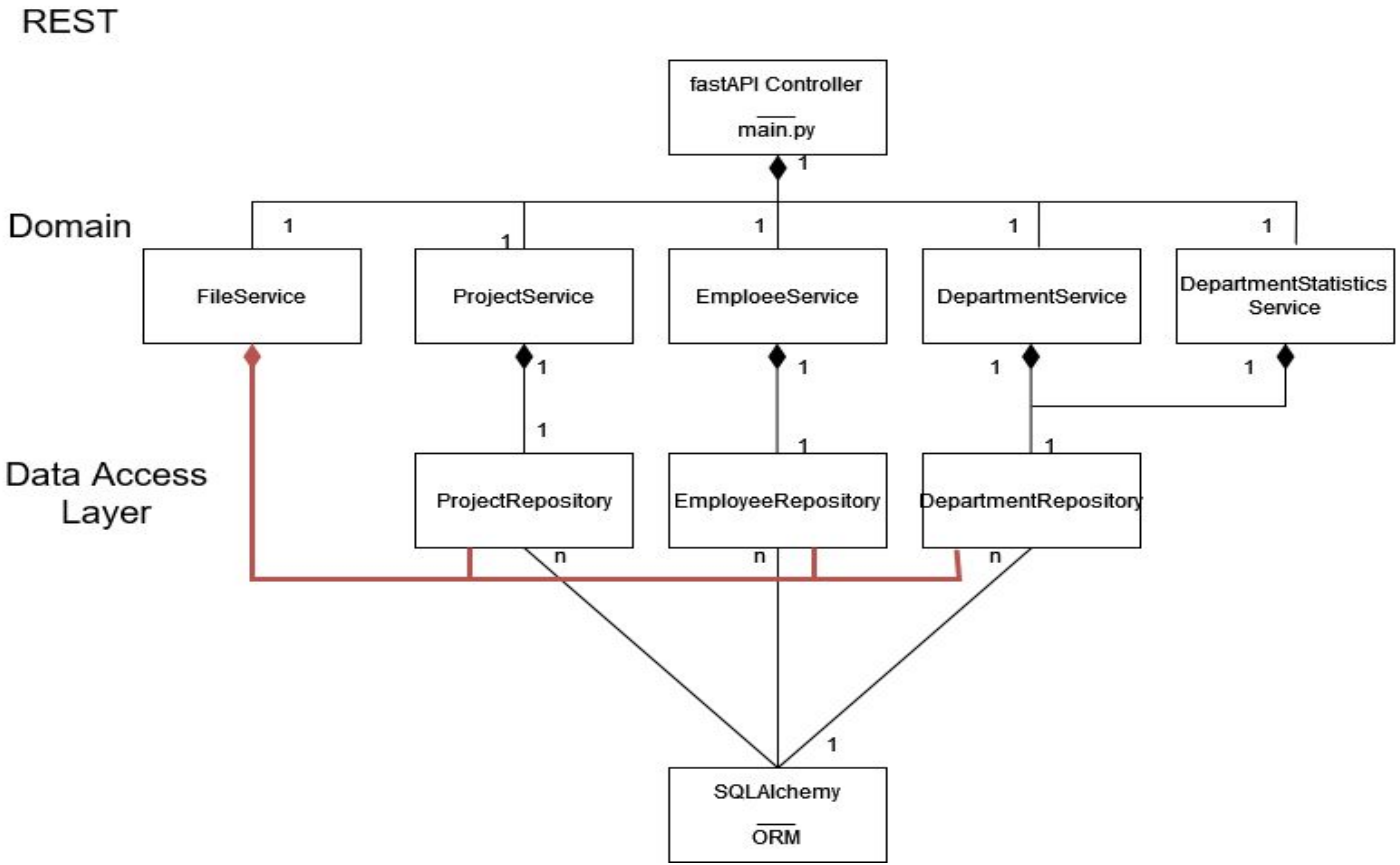
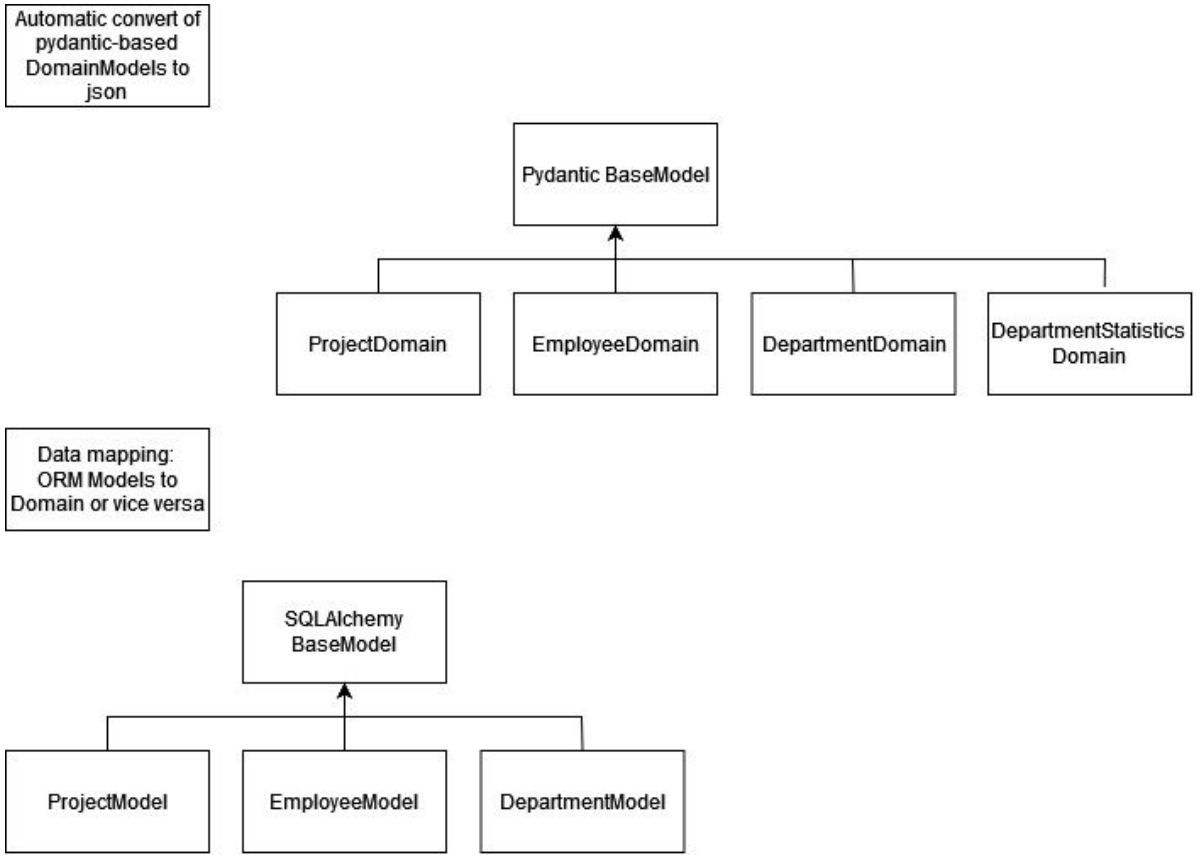
Architektur



Architektur: Datenrepräsentation



Wahl des WebFrameworks

- Kandidaten:
  - Django => “Overkill”
  - Flask => ORM Wrapper, REST Wrapper schwierig
  - fastAPI
  - pyramid => steile Lernkurve, viele Freiheiten
- Wahl:
  - fastAPI
- Warum?
  - hervorragende DX
  - gut dokumentiert
  - wenig Overhead
  - pydantic Support out-of-the-box
  -

Persistenz

- 1NF => atomare Attribute
- 2NF => keine partiellen Abhängigkeiten
- 3NF => keine transitiven Abhängigkeiten von Nichtschlüsseln zum Primärschlüssel

- Über SQLAlchemy in SQL abbildbar
- Employee PK suboptimal
  - -sensibles Datum, nicht auf API exposen
  - Performance-hit bei Indezierung, Joins
  - Mailänderung bedeutet FK Change

Verbesserungsvorschläge

- Clean Architecture:
  - divide et impera: eigene Files für Services, Repositories, Spaltung des Controllers
  - Interfaces statt tight coupling, DI einführen
  - ggf. Microservicing?
- Clean Code:
  - fastAPI unterstützen, Documentation anhängen!
  - Config-Parameter in ENV
  - Graceful exception handling
  - UnitTests, IntegrationTests, End-to-End-Tests…
- Funktionalität:
  - App dockerizen, Deployment mit docker-compose
  - I/O-heavy Methoden async