ANALISI TECNICA

Nome Progetto	Gestione Conto Corrente
Cliente	Mario Rossi Banking
Versione Documento	1.0
Autore	Daniele Cocuzza
Data Creazione	23/12/2022

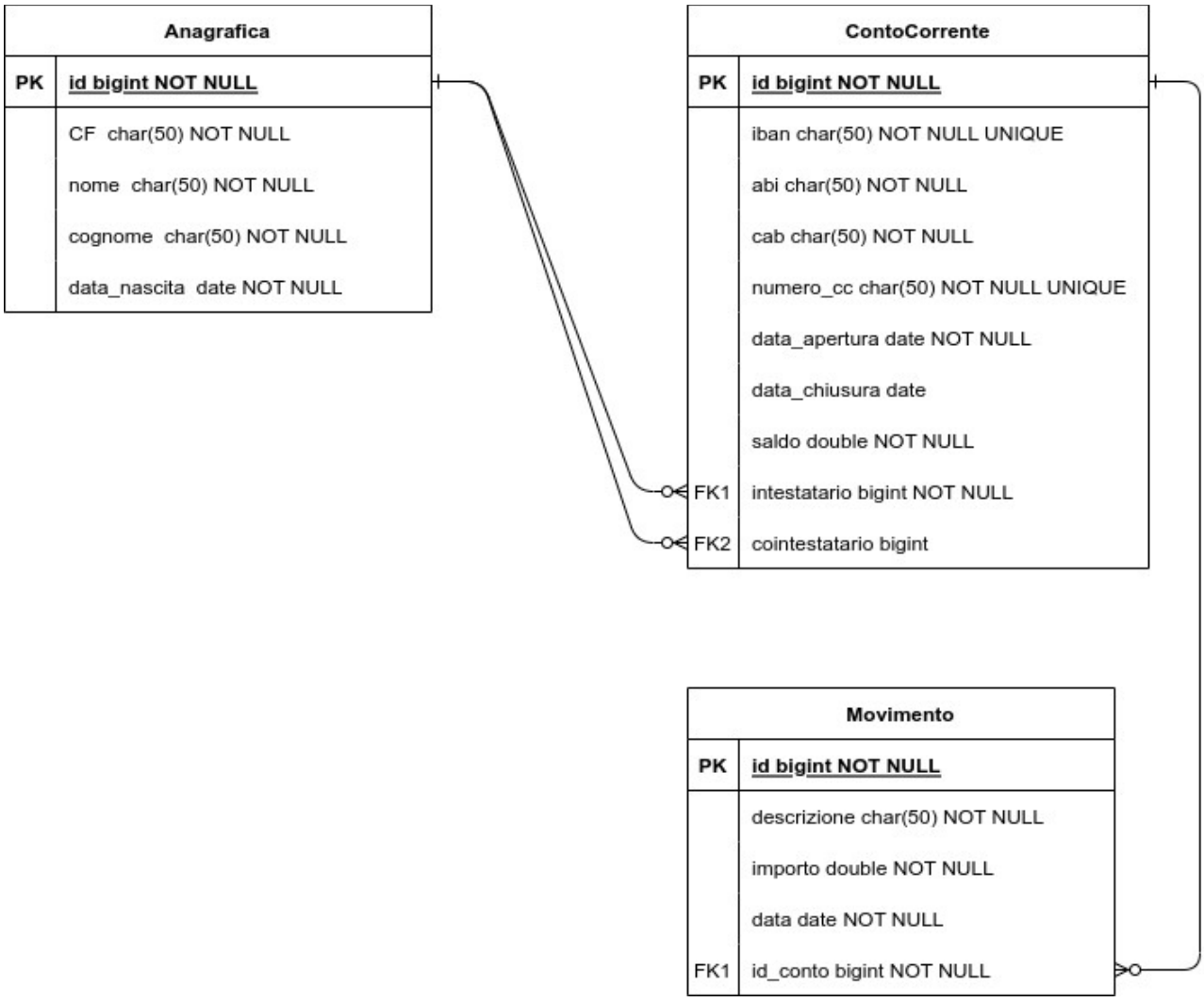
Business Requirement

Crea un programma per la gestione di un conto corrente. Il programma deve essere in grado di effettuare versamenti, prelievi, di restituire il saldo e di estrarre la lista degli ultimi 5 movimenti.

Tecnologie di Implementazione

Linguaggio	Java 18
Framework	Spring Boot
Comunicazione HTTP	API RESTful
Database	MySQL

Diagramma E-R



Repository

Si utilizzerà l'interfaccia **JpaRepository** per l'interazione con il Database.
Tutti i Repository saranno annotati con **@Repository**
Verranno creati i seguenti Repository i quali conterranno le operazioni CRUD:

- **AnagraficaRepository**
- **ContocorrenteRepository**
- **MovimentoRepository**

Per l'implementazione dei metodi richiesti nell'AFU:

MovimentoRepository

Prelievo: **List<double> prelievo(Long id_conto, double importo)** conterrà la query per permettere il prelievo di denaro.

Versamento: **void versamento(Long id_conto, double importo)** conterrà la query per permettere il versamento di denaro.

Ultimi 5 movimenti: **List<Movimento> getMovimentiById(Long id_conto)** restituisce i movimenti specificati nell'id.

ContocorrenteRepository

Saldo: **List<double> getSaldo(Long id)** conterrà la query per ritornare il saldo del conto corrente specificato dall'id.

Service

I Service saranno:

- **AnagraficaService**
- **ContocorrenteService**
- **MovimentoService**

Tutti i Service saranno annotati con **@Service**
I Service avranno un riferimento al rispettivo Repository tramite Dependency Injection.

Metodi di base
Ogni Service avrà i metodi per le operazioni CRUD, tra cui:

- Inserimento: **void add(Object o)**
- Lettura: **Optional<Object> getById(Long id)**
- Cancellazione: **voidDeleteById(Long id)**
- Aggiornamento: **update(Long id, Object o)**

I Service **ContocorrenteService** e **MovimentoService** oltre ai metodi per le operazioni CRUD avranno i metodi per la gestione delle operazioni richieste nell'AFU

Implementazione dei metodi richiesti nell'AFU:

MovimentoService

Prelievo: **List<double> prelievo(Long id_conto, double importo),** il metodo ritorna **movimentoRepository.prelievo(Long id_conto, Double importo)** implementato su MovimentoRepository.

Versamento: **void versamento(Long id_conto, double importo)** il metodo richiama **movimentoRepository.versamento(Long id_conto, Double importo)** implementato su MovimentoRepository

Ultimi 5 movimenti: **List<Movimento> getMovimentiById(Long id_conto)** il metodo prende i movimenti da **movimentoRepository.getMovimentiById(Long id_conto)** e filtra gli ultimi 5 movimenti.

ContocorrenteService

Saldo: **List<double> getSaldo(Long id**), il metodo ritorna **contocorrenteRepository.getSaldo(Long id)** implementato su ContocorrenteRepository.

Controller

I Controller permettono la gestione delle chiamate API, saranno implementati i seguenti Controller:

- **AnagraficaController**
- **ContocorrenteControlle**r
- **MovimentoController**

I Controller saranno annotati con **@RestController**
Per i vari endpoint consultare il seguente link:
Documentazione API

Stime

Tipologia intervento	GG/u	Note
Studio di fattibilità	1,5	
Controller	0,5	Non è stata prevista ancora una parte di sicurezza in questa versione
Service	1	
Repository	0,5	
Entity	0,5	
Totale Sviluppo Java	2,5	
Sviluppo Database	0,5	
Test unitari	0,5	
TOTALE	5	