**Títol:** Projecte "MiniBank" — backend en Java amb evolució cap a persistència SQL

**Objectiu:** Crear un projecte orientat al backend i a la persistència de dades en Java per poder escalar-lo progressivament i, alhora, aprendre a programar en Java i entendre les seves particularitats. El projecte es publicarà a GitHub amb una explicació pas a pas i una documentació exhaustiva en Markdown (README i fitxers addicionals).

**Estructura inicial (versió 1):**

* Arquitectura: Model — Vista — Controlador (MVC) bàsica.
* Gestió del projecte: al inici cap però evolucionant a Maven que més endavant migrarà a Spring Boot.
* Tests: JUnit inclòs en les versions de Maven.
* Persistència: En la primera fase s’emmagatzemarà la informació en arrays o HashMap dins el codi; posteriorment es migrarà a una base de dades SQL.

**Classes principals (4):**

Customer — dades rellevants del client i llista d'accounts associades. private final int idCus;  
private String name;  
private String firstLastName;  
private String secondLastName;  
private final String nif;  
private final LocalDate dateBirth;  
private final String sex;  
private String address;  
private String zipCode;  
private String city;

private List<Account> accounts;  
private static int idCustomer = 1;

1. Account — dades rellevants del compte; mètode per llistar totes les transaccions relacionades; llista de préstecs (Loan) vinculats al compte; mètode per validar si es pot concedir un préstec abans de crear-lo.

private final int idAccount;

private final String accountNumber;

private final LocalDate openingDate;

private float balance;

private boolean activa;

private static int id = 1;

private List<Transaction> transactions;

private List<Loan> loans;

1. Transaction — transaccions més comunes (retirament i dipòsit) amb les dades principals. Haurà de tenir en compte l’execució de préstecs: si hi ha un préstec concedit, caldrà generar els moviments mensuals (per exemple, el retirament per la quota de retorn acordada i l’operació inicial de dipòsit del capital del préstec).

private final Type type;

private final float amount;

private final LocalDateTime date;

private final float balanceAfter; // saldo tras la operación

public enum Type { DEPOSIT, WITHDRAWAL, LOANDEPODSIT, LOANWITHDRAWL}

1. Loan — especificacions del préstec (import, termini, tipus d’interès, calendari de pagaments, estat, etc.).

private final int idLo;

private int IdLo;

private final float amount; // importe del préstamo

private final float interest; // interés anual (%)

private final LocalDate startDate;

private final LocalDate dueDate;

private float remainingBalance; // saldo pendiente

private boolean active;

private static int idLoan = 1;

private ArrayList<Transaction> transac

**Funcionalitats inicials (mètodes bàsics):**

1. Crear un client amb les dades més rellevants i la seva llista de comptes.
2. Crear comptes amb les dades rellevants, llistar transaccions, i vincular préstecs als comptes; validar condicions per concedir préstecs.
3. Registrar transaccions: dipòsit i retirada; gestionar pagaments recurrents derivats de préstecs segons les condicions establertes.
4. Crear i gestionar objectes Loan amb la seva lògica (càlcul de quotes, calendari, estat, amortitzacions).

**Fases del projecte / passos de treball:**

1. Definir l’esquema SQL i generar l’script de creació de taules (diagrama entitat-relació). Aquest script será la guia inicial i s’actualitzarà segons es requereixi.
2. A partir de l’ERD o l’script SQL, generar UML per a les classes Java i les relacions.
3. Implementar les classes i mètodes de manera iterativa; versionar cada canvi en un repositori GitHub, escrivint commits descriptius que expliquin els passos realitzats.

**1) Versió corregida (català)**

**Títol:** Projecte "MiniBank" — backend en Java amb evolució cap a persistència SQL

**Objectiu:** Crear un projecte orientat al backend i a la persistència de dades en Java per poder escalar-lo progressivament i, alhora, aprendre a programar en Java i entendre les seves particularitats. El projecte es publicarà a GitHub amb una explicació pas a pas i una documentació exhaustiva en Markdown (README i fitxers addicionals).

**Estructura inicial (versió 1):**

* Arquitectura: Model — Vista — Controlador (MVC) bàsica.
* Gestió del projecte: al inici sense eina concreta, però evolucionant cap a **Maven** i més endavant migrarà a **Spring Boot**.
* Tests: JUnit inclòs en les versions de Maven.
* Persistència: En la primera fase s’emmagatzemarà la informació en arrays o HashMap dins del codi; posteriorment es migrarà a una base de dades SQL.

**Classes principals (4):**

**Customer**

private final int idCus;

private String name;

private String firstLastName;

private String secondLastName;

private final String nif;

private final LocalDate dateBirth;

private final String sex;

private String address;

private String zipCode;

private String city;

private List<Account> accounts;

private static int idCustomer = 1;

**Account**

private final int idAccount;

private final String accountNumber;

private final LocalDate openingDate;

private float balance;

private boolean active;

private static int id = 1;

private List<Transaction> transactions;

private List<Loan> loans;

**Transaction**

private final Type type;

private final float amount;

private final LocalDateTime date;

private final float balanceAfter; // saldo després de l'operació

public enum Type { DEPOSIT, WITHDRAWAL, LOAN\_DEPOSIT, LOAN\_WITHDRAWAL }

**Loan**

private final int idLo;

private final float amount; // import del préstec

private final float interest; // interès anual (%)

private final LocalDate startDate;

private final LocalDate dueDate;

private float remainingBalance; // saldo pendent

private boolean active;

private static int idLoan = 1;

private List<Transaction> transactions;

**Nota:** s'han corregit noms i tipus, i s'ha normalitzat la nomenclatura (per exemple active en comptes de activa, LOAN\_DEPOSIT/LOAN\_WITHDRAWAL en l'enum, etc.).