1. Download required Tools
   1. Intelij Community
   2. Postgres
   3. DbEaver Community
   4. Keycloak
   5. Node.js
   6. Angular
   7. Java 8 JDK (includes JRE, JVM)
   8. Maven (Binary zip archive)
   9. Git
2. Setup Path Variables for each bin
   1. Intelij
   2. Node js
   3. Postgres
   4. Maven
   5. Git
3. Check Setup
   1. npm --v
   2. node -v
   3. java -version
   4. postgres -V
   5. ng version
   6. mvn -version
   7. git -v
4. Setup Intelij with JDK and Maven
5. Start a project: start.spring.io
   1. Dependencies
      1. Spring Web
      2. Spring Data Jpa
      3. H2 Database
      4. Lombok
      5. Spring Boot DevTools
   2. Project: Maven Project
   3. Language: Java
   4. Spring Boot: 2.6.10
   5. Packaging: Jar
   6. Project Metadata: …
6. Open Project In intelij and add SonarLit Plugin
7. Build Project To check everything is ok
8. Initial commit on github
   1. Git init // default by Intelij
   2. Git checkout -b dev
   3. Git add.
   4. Git commit -m ‘Initial Commit’
   5. Git remote add origin URLINK
   6. Git push
   7. git push --set-upstream origin dev
9. From Application Package create
   1. entities
   2. repositories
   3. services
   4. web
   5. dto
   6. mappers
   7. enums
10. Add entities
    1. AccountOperation
    2. BankAccount
    3. CurrentAccount
    4. Customer
    5. SavingAccount
    6. Referential
11. Add Mapping Object Relational
    1. @Entity
    2. @Id @GeneratedValue (strategy=GenerationType.Identity)
    3. @OneToMany(mappedBy=”xxx”, FetchType.Lazy), @ManyToOne
    4. Tips: list side is the one to many
    5. @Inheritance (strategy = InheritanceType.SINGLE\_TABLE)
    6. @DiscriminatorColumn (name=” TYPE”, length=4)
    7. @DiscriminatorValue
    8. @Enumerated(EnumType.STRING)
    9. @JsonProperty(access = JsonProperty.Access.WRITE\_ONLY)
12. Specify Database on application.properties
    1. Spring.datasource.url=jdbc:h2:mem:bank
    2. Spring.h2.console.enabled=true
    3. Server.port=8085
13. Create required repositories
    1. Interface => extends JpaRepository <entity,Long>{}
14. Test persistence in Database
    1. Application
    2. @Bean
    3. CommandLineRunner start(Repository repo){
    4. Return args ->{
    5. Class class = new Class();
    6. Class.setx(‘hello’);
    7. Repository.save(class);});};}
15. If everything looks good switch to postgres
    1. From pom.xml comment h2 dependency
    2. Add postgres dependency
    3. Application.properties change h2 with postgres
    4. spring.datasource.url=jdbc:postgresql://localhost:5432/bank
    5. spring.datasource.username=postgres
    6. spring.datasource.password=postgres
    7. spring.datasource.driver-class-name=org.postgresql.Driver
    8. spring.jpa.show-sql=true
    9. spring.jpa.hibernate.ddl-auto=create-drop
    10. spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.PostgreSQLDialect
16. Start Postgres server from postgres directory
    1. pg\_ctl -D data start
17. Go to Dbeaver and setup connection to postgres server
18. IF everything looks good
19. Service create service
    1. Nouvelle interface qui contient le besoin fonctionnel (Déclaration des méthodes)
    2. Créer le service qui contient l’implémentation de l’interface ;
    3. Add annotation @Service, @Transactional
    4. Import requiered repositories and injection de dépendance using @AllArgsConstructor
20. Add slf4j annotation on serviceImpl
    1. @Slf4j
    2. And use it like log.info(‘Hello World’);
21. New package exceptions
    1. NotFoundException extends RuntimeException
    2. String message
    3. Super(message)
22. Back to service and add throws the not found exception
23. Once All services are done
24. We copy paste entity to DTO
25. Then we create Mapper imlplimentation
    1. Add notation service
    2. Using Mapstruct