# Subtask 4; Dokumentáció készítése /know-how

## 4/1; Adatbázis migráció?

**Forrás: Copilot**

**Az adatbázis migráció egy olyan folyamat, amely során az adatbázis sémáját vagy adatait egy adott állapotból egy másik állapotba változtatjuk.** Ez magában foglalhatja az új táblák létrehozását, meglévő táblák módosítását, adatok áthelyezését vagy törlését. Az adatbázis migrációk célja, hogy az adatbázis struktúráját és tartalmát szinkronban tartsák az alkalmazás változásaival. Az adatbázis migrációk kezelése érdekében gyakran használnak eszközöket, mint például a Flyway vagy a Liquibase, amelyek lehetővé teszik a migrációk verziókezelését és automatizálását. Ezek az eszközök segítenek abban, hogy a fejlesztők könnyen alkalmazhassák a szükséges változtatásokat az adatbázisban, és biztosítsák, hogy minden környezetben (fejlesztői, teszt, éles) ugyanazok a változtatások legyenek alkalmazva. Például egy Flyway migrációs fájl így nézhet ki:

```
-- V1__init.sql
CREATE TABLE application_user (
   id BIGSERIAL PRIMARY KEY,
   active BOOLEAN,
   default_course_id BIGINT,
   email VARCHAR(255),
   first_name VARCHAR(255),
   last_name VARCHAR(255),
   . . .
);
```

Ez a migrációs fájl létrehoz egy application_user nevű táblát az adatbázisban a megadott oszlopokkal.

## 4/2; Flyway?

A Flyway egy hatékony eszköz az adatbázis migrációs eszköz a fejlesztők részére az adatbázis + adat változtatások nyomon követésében és automatizálásában.

**A Flyway egy adatbázis migrációs eszköz, részben** nyílt forráskódú, segít sokféle adatbázis rendszerbeli adatbázis-séma módosítások kezelésében, nyomon követésében és telepítésében. Java-orientált, és Java-alapú migrációt biztosít, Java alapú, és széles körű támogatást nyújtanak az olyan Java keretrendszerekhez, mint a Spring Boot és a Vert.x. Támogatja az olyan builderekkel való integrációt, mint a Maven és a Gradle.

- Verzióalapú migrációs megközelítést nyújt az adatbázis-séma módosításához.
- A megadott .sql szkripteken keresztül,parancssorból függetlenül futtatható, tehát **SQL-t használ a változás meghatározásához.**
- **A Flyway** egy lineáris adatbázis-verziórendszerre épül, amely minden egyes verzióváltásnál növekszik. Ez néha konfliktusokat okozhat a párhuzamos fejlesztéssel. Az .sql szkript fájlneve határozza meg a migráció típusát általánosságban így:
    - V01__Add_New_Column.sql
        - V (a versionált), U (a visszavonás) és R (a megismételhető), utána verziószám
- A **Flyway** a migrációkat az adatbázis sémájában tárolja alapértelmezetten a **flyway_schema_history** nevű táblában. Postgres adatbázisban a public schema-ban.
- A változás sorrendjének kezelése viszonylag nehéz a **Flywayben**. A **Flyway** esetében a sorrend a fájlnévben szereplő verziószámtól és a migráció típusától függ.

- A visszaállításra (rollback), ha egy rossz módosítás katasztrofális problémát okozott az alkalmazásban: a **Flyway** rendelkezik egy visszavonó migrációval is, amelyet egy U-val kezdődő fájlnévvel lehet telepíteni, amelyet a visszavonandó verzió követ, ingyenes verziója megfelelően jó, de természetesen a fizetős verziója még összetettebb visszavonási funkcionalitást is kínál.
- A **Flyway** nehézkes környezetenként mást és mást telepíteni (vagy nem telepíteni), mert akkor minden környezet/adatbázishoz más konfigurációs fájlt kellene beállítani.
- Java-alapú migrációt biztosít.
- A **Flyway** nem támogat előfeltételeket, de az egyes sql-procedúrákon keresztül a legtöbb SQL-alapú adatbázisban írhatunk.

# Subtask 5; Prezentálás a fejlesztői csapat számára /know-how

A prezentáció célja, hogy a csapat minden tagja értse és átlássa a Flyway használatát egy konkrét appban és szükség esetén tudja használni.

## 5/1; Bevezetés az adatbázis migrációba

**Forrás: Copilot**
**Az adatbázis migráció egy olyan folyamat, amely során az adatbázis sémáját vagy adatait egy adott állapotból egy másik állapotba változtatjuk.** Ez magában foglalhatja az új táblák létrehozását, meglévő táblák módosítását, adatok áthelyezését vagy törlését. Az adatbázis migrációk célja, hogy az adatbázis struktúráját és tartalmát szinkronban tartsák az alkalmazás változásaival. Az adatbázis migrációk kezelése érdekében gyakran használnak eszközöket, mint például a Flyway vagy a Liquibase, amelyek lehetővé teszik a migrációk verziókezelését és automatizálását. Ezek az eszközök segítenek abban, hogy a fejlesztők könnyen alkalmazhassák a szükséges változtatásokat az adatbázisban, és biztosítsák, hogy minden környezetben (fejlesztői, teszt, éles) ugyanazok a változtatások legyenek alkalmazva. Például egy Flyway migrációs fájl így nézhet ki:

## 5/2; Flyway használata általánosságban

A Flyway egy nyílt forráskódú adatbázis-migrációs eszköz, amely segít az adatbázis sémájának és adatainak verziókezelésében és automatizálásában, főbb jellemzői:

- **Verziókezelés**: az adatbázis sémáját és adatait verziókezeli, így könnyen nyomon követhetők a változtatások.
- **Automatizálás**: az adatbázis migrációk automatizálhatók, így biztosítva, hogy minden környezetben (fejlesztői, teszt, éles) ugyanazok a változtatások legyenek alkalmazva.
- **Egyszerű integráció:** könnyen integrálható különböző build eszközökkel (pl. Maven, Gradle) és CI/CD rendszerekkel.
- **Többféle adatbázis-kezelő rendszer támogatása:** PostgreSQL, MySQL, Oracle, SQL Server stb.

- **SQL és Java migrációk**: támogatja az SQL és Java alapú migrációkat is, így rugalmasan használható különböző igényekhez.

**Alapvető használat:**

- **1; Migrációs fájlok létrehozása**: egy névkonvenció szerint pl. V1__init.sql, V2__add_table.sql
- **2; Migrációk futtatása**: a Flyway automatikusan felismeri és végrehajtja a még nem alkalmazott migrációkat az adatbázisban.
- **3; Konfiguráció**: a Flyway konfigurálható különböző paraméterekkel, például az adatbázis URL, felhasználónév, jelszó stb.

**Egy Flyway migrációs fájl:**

```
-- V1__init.sql
CREATE TABLE application_user (
   id BIGSERIAL PRIMARY KEY,
   active BOOLEAN,
   default_course_id BIGINT,
   email VARCHAR(255),
   first_name VARCHAR(255),
   last_name VARCHAR(255),
   . . .
);
```

**Flyway konfiguráció egy `application.yaml` fájlban:**

```
spring:
  flyway:
   baseline-on-migrate: true
   locations: classpath:db/migration
   out-of-order: true
```

**Előnyök:**
- **Könnyű használat**: Egyszerűen használható és gyorsan bevezethető a projektekbe.
- **Biztonság**: Biztosítja, hogy az adatbázis mindig konzisztens állapotban legyen.
- **Rugalmasság**: Támogatja a különböző adatbázis-kezelő rendszereket és migrációs típusokat.

## 5/3; Flyway használata egy Spring Boot appban

- **1; pom.xml-be flyway dependencia hozzáadása**

```
<dependency>
   <groupId>org.flywaydb</groupId>
   <artifactId>flyway-core</artifactId>
</dependency>
```

- **2; app.yaml flyway konfigurációs beállításai:**

```
spring:
 flyway:
```

```
      baseline-on-migrate: true
      locations: classpath:db/migration
      out-of-order: true
```

- **3; Migrációs fájlok létrehozása a src/main/resources/db/migration könyvtárban:**

```
-- V1__init.sql
CREATE TABLE application_user (
   id BIGSERIAL PRIMARY KEY,
   active BOOLEAN,
   default_course_id BIGINT,
   email VARCHAR(255),
   first_name VARCHAR(255),
   last_name VARCHAR(255),
   . . .
);
```

- **4; adatbázis konfig check in app.yaml fájlban:**

```
spring:
 datasource:
   url: jdbc:postgresql://localhost:5433/database_name
   username: postgres
   password: root
   driver-class-name: org.postgresql.Driver
 jpa:
   hibernate:
     ddl-auto: validate
```

- **5; alkalmazás futtat, és a Flyway automatikusan végrehajtja a migrációkat az adatbázisban.**

- `baseline-on-migrate: true` ha egy üres adatbázisra futtatjuk a migrációkat, a Flyway létrehoz egy alapvonalat (baseline) a migrációk előtt. Ez hasznos lehet, ha egy már létező adatbázist szeretnénk migrációs eszközzel kezelni, és el akarjuk kerülni a már meglévő sémák újraalkalmazását.
- `locations: classpath:db/migration` meghatározza, hogy hol találhatók a migrációs fájlok. A classpath:db/migration azt jelenti, hogy a migrációs fájlokat a projekt classpath útvonalán belül a db/migration könyvtárban (ez az src/main/resources/db/migration könyvtár lesz) kell elhelyezni. A Flyway innen fogja betölteni és végrehajtani a migrációs fájlokat.
- `out-of-order: true` lehetővé teszi, hogy a migrációk nem sorrendben kerüljenek végrehajtásra. Ha egy korábbi verziójú migrációs fájl később kerül hozzáadásra, mint a már végrehajtott migrációk, a Flyway végrehajtja ezt a migrációt is. Ez hasznos lehet, ha több fejlesztő dolgozik párhuzamosan és különböző migrációkat hoznak létre.

**classpath:**

A projekt osztályútvonala (classpath) az a hely, ahol a Java futtatókörnyezet keresi az osztályfájlokat és egyéb erőforrásokat. Maven alapú Java projektekben ezen könyvtárak tartoznak az classpath-hoz
:

- `src/main/java` - Az alkalmazás forráskódja.
- `src/main/resources` - Az alkalmazás erőforrásai (pl. konfigurációs fájlok, statikus erőforrások).
- `target/classes` - A lefordított osztály fájlok.

A Flyway migrációs fájlok esetében a classpath:db/migration azt jelenti, hogy a migrációs fájlokat a src/main/resources/db/migration könyvtárban kell elhelyezni.

**gererate_changelog.sh fájl:**

- ő egy futtatható bash script
- ezzel a kis bash szkripttel hozzuk létre az új üres tartalmú .sql scripteket
- **futtatása**

- például:(ajánlott a resources/db> mappából) `generate_changelog.sh 8 34`
- ahol a 8 a sprint száma, a 34 pediga ticket száma
- lép be a cmd-édbe (ajánlott a Run As admin)
- `./generate_changelog.sh 18 240`
- **app-backend-team1\src\main\resources\db> generate_changelog.sh 18 240**

C:\Windows\system32>cd C:\Users\HP\IdeaProjects\app-backend-team1\src\main\resources\db
C:\Users\HP\IdeaProjects\app-backend-team1\src\main\resources\db>generate_changelog.sh 18 240

```
# Azért, hogy ne legyenek ütközések, és egységesen tudjunk kezelni migrációs
# szkripteket, ezzel a kis szkripttel tudunk létrehozni újat, ha szükséges
#
# Egyszerűen csak adjuk át paraméterként a sprint és a ticket számát
# Kiadható többször egymás után, hiszen a másodperc is a fájlnév része,
# és így lesz folytatólagos
#
# pl.:
# ./generate_changelog.sh 8 34

SPRINT_NUMBER=$1
TICKET_NUMBER=$2
DATE_TIME=$(date '+%Y_%m_%d_%H_%M_%S')

if [ "$SPRINT_NUMBER" != "" ] || [ "$TICKET_NUMBER" != "" ]; then

    MIGRATION_SCRIPT_IDENTIFIER= V"$SPRINT_NUMBER"_$(echo $DATE_TIME)__FBA-$2

    NEW_SCRIPT_FOLDER=./migration/"S$1"

 mkdir -p $NEW_SCRIPT_FOLDER
 touch $NEW_SCRIPT_FOLDER/$MIGRATION_SCRIPT_IDENTIFIER.sql
 echo "Created: $NEW_SCRIPT_FOLDER/$MIGRATION_SCRIPT_IDENTIFIER.sql"

else
 echo "Missing parameters!"
fi
```

**Ez a bash script a következő lépéseket hajtja végre:**

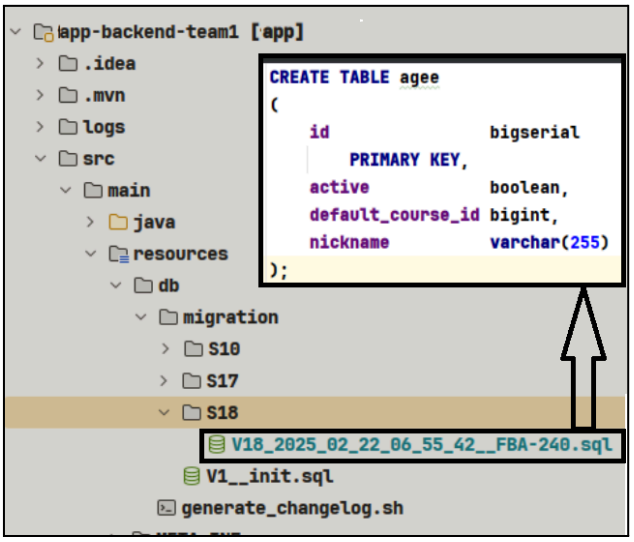
- 1. Beállítja a `SPRINT_NUMBER` változót `8`-ra és a `TICKET_NUMBER` változót `34`-re.
- 2. Létrehozza a `DATE_TIME` változót az aktuális dátum és idő formátumával: `YYYY_MM_DD_HH_MM_SS`.
- 3. Összeállítja a `MIGRATION_SCRIPT_IDENTIFIER` változót a következő formátumban: `V8_YYYY_MM_DD_HH_MM_SS__FBA-34`.
- 4. Létrehozza a `NEW_SCRIPT_FOLDER` változót a következő formátumban: `./migration/S8`.
- 5. Létrehozza a `NEW_SCRIPT_FOLDER` könyvtárat, ha még nem létezik.
- 6. Létrehoz egy üres SQL fájlt a `NEW_SCRIPT_FOLDER` könyvtárban a `MIGRATION_SCRIPT_IDENTIFIER` névvel.
- 7. Kiírja a létrehozott fájl nevét a konzolra.

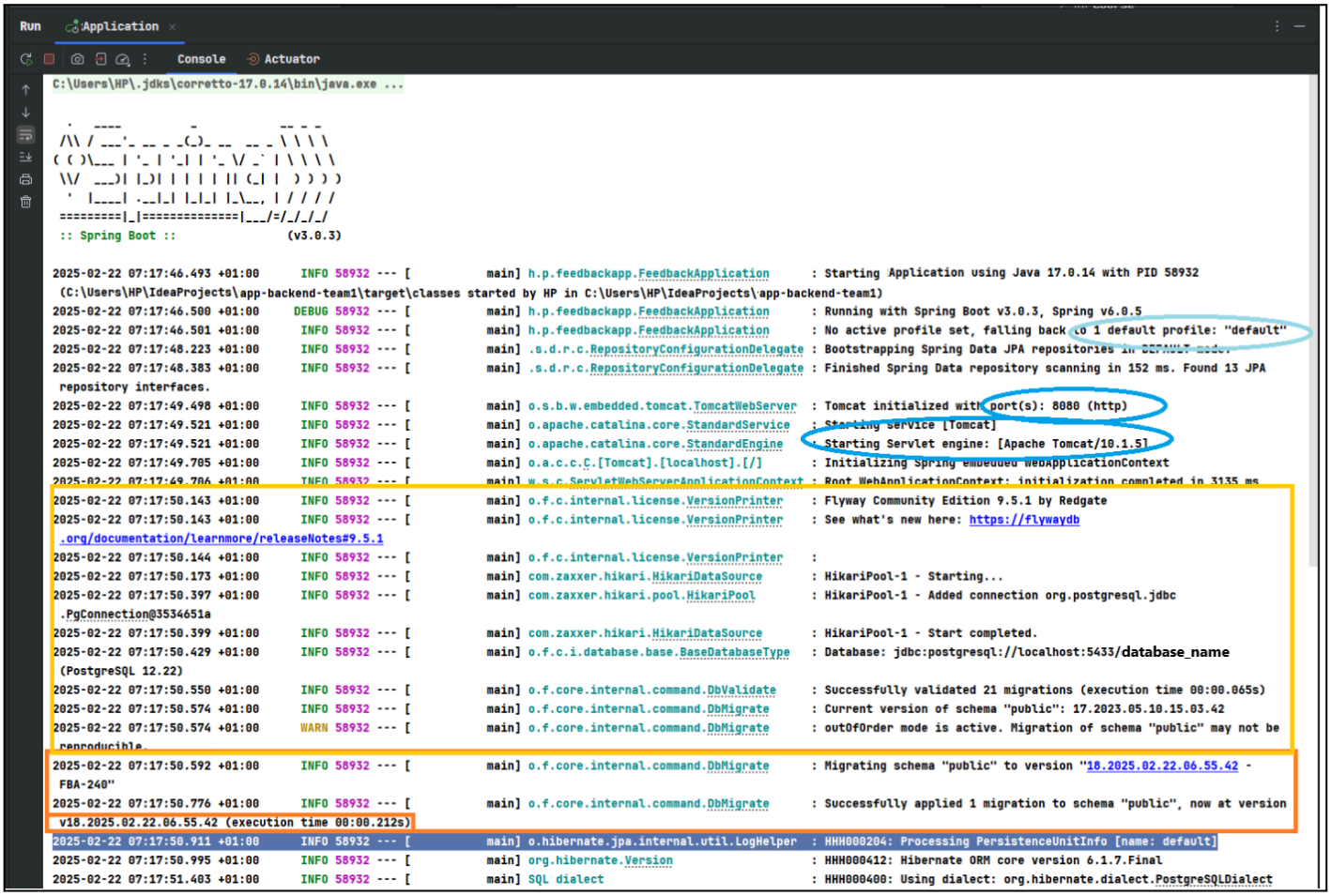
Például, ha a script 2023. október 10-én, 15:30:45-kor fut, a következő fájl jön létre:

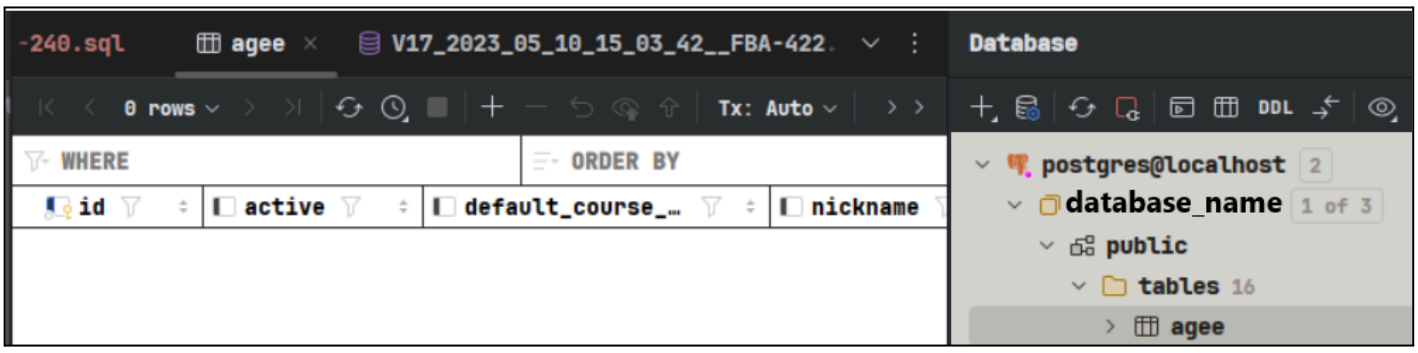
./migration/S8/V8_2023_10_10_15_30_45__FBA-34.sql

Feltöltöd a futtatandó sql parancsodat, ami bármi adatbázis építő lehet: pl. CREATE TABLE, INDEX, vagy INSERT, UPDATE… stb.

Ezek után elindítd a Spring Boot alkalmazásodat, és a Spring Boot automatikusan lefuttatja az eddig még nem lefutott, tehát az új .sql scriptedet.



# 5/4; Jövőbeli használat útmutató

A domain/ -be tett JPAs'@Entity-eden változtatsz, pl. hozzáadsz a .java fájl forráskódban egy új mezőt, akkor a Flyway-el is ezt fel kell szedetni! Lépések:

**1; Entitás Módosítása**
Módosítsd a Java entitás osztályt a szükséges változtatásokkal. Például, ha az ApplicationUser entitáshoz hozzáadunk egy új mezőt:

```
@Entity
public class ApplicationUser {
   @Id
   @GeneratedValue(strategy = GenerationType.IDENTITY)
   private Long id;
   private Boolean active;
   private Long defaultCourseId;
   private String email;
   private String firstName;
   private String lastName;
   . . .
   private String newField; // Új mező hozzáadása

   // Getters és Setters
}
```

**2; Flyway Migrációs Fájl Létrehozása**
Hozz létre egy új Flyway .sql migrációs fájlt a src/main/resources/db/migration könyvtárban. Például

**2/1; generate_changelog.sh val hozd létre az üres sql script fájlodat**

- CMD-ét ha lehet Admin userként nyisd meg
- C:\Windows\system32>cd C:\Users\HP\IdeaProjects\app…\src\main\resources\db
- C:\Users\HP\IdeaProjects\app…\src\main\resources\db>generate_changelog.sh 18 241

**2/2; Töltsd fel a megfelelő, az imént sql utasítással**

```
-- V2__add_new_field_to_application_user.sql
ALTER TABLE application_user ADD COLUMN new_field VARCHAR(255);
```

**3. Flyway Konfiguráció Ellenőrzése**
A Flyway megfelelően legyen konfigurálva az application.yaml fájlban:

```
spring:
 flyway:
   baseline-on-migrate: true
   locations: classpath:db/migration
```

**4. Alkalmazás Futtatása**
Indítsd el az alkalmazást, és a Flyway automatikusan végrehajtja az új migrációs fájlt, amely hozzáadja az új mezőt az adatbázis táblához.  Ezekkel a lépésekkel sikeresen módosíthatod az entitást és az adatbázis sémát Flyway használatával.