

Alkalmazás fejlesztések GY.

# Classroom Application

BACKEND DOKUMENTÁCIÓ

Foltán Dániel  
LC5K0C  
foltandani@gmail.com

Albirt Mátyás  
S6ACON  
amatyi96@gmail.com

Bárkányi Dominik  
TIFYTA  
barkanyi96@gmail.com

# Tartalomjegyzék

I.	<b>Fejlesztői környezet.....</b>	Hiba! A könyvjelző nem létezik.
	Bemutatása	
	Beállítása	
	Használt technológiák	
II.	<b>Adatbázis-terv.....</b>	2
	UML diagram	
III.	<b>Könyvtárstruktúra.....</b>	Hiba! A könyvjelző nem létezik.
	Bemutató	
IV.	<b>Végpont-tervek.....</b>	Hiba! A könyvjelző nem létezik.
	Leírások	
	1_db végpont működésének leírása(szekvenciadiagram	

# Fejlesztői környezet

## Bemutató

Az applikáció fejlesztése Windows 10 -es rendszeren történik és IntelliJ IDEA fejlesztői környezetet használunk, amely egyetemi hallgatók részére ingyenesen elérhető. Érhető és könnyű kezelése mellett nagy segítséget nyújt akár másféle program elkészítésében is.

## Beállítás

Spring Boot projektünket a Spring Initializer(<https://start.spring.io/>) eszközzel készítjük el, ahol a következő beállításokat használjuk:

- Felül a lenyíló listákban:
  - maven
  - java
  - 1.5.7
- Group: hu.elte.alkfejl
- Artifact: classroomApplication

## Használt technológiák

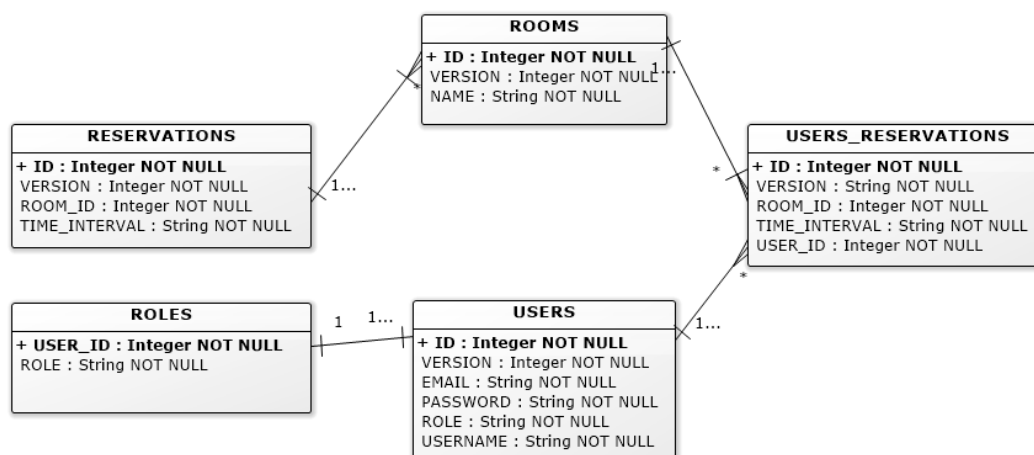
A projektünkhöz több dependencies-t is használunk segítségképpen:

- H2 - in memory database
- JPA - spring-data-jpa -> repositoryk használatához
- Lombok - automatikus getter, setter, constructor generálás
- Thymeleaf - nézet generálása
- DevTools - fejlesztéshez segítség

Beállítások után letölthető a Spring Boot projektünk, amelyet IntelliJ IDEA-ban megnyitva fejleszthetünk is.

# Adatbázis-terv

## UML diagram



# Könyvtárstruktúra

---

## Bemutató

Projektünkben MVC architektúrát valósítunk meg, ami egy olyan struktúra, amely segítségével a szoftverünk szerepkeit szétválasztjuk, az egyes rétegeket egymástól külön kezeljük.

A struktúra megtartásával az egyes rétegek könnyebben cserélhetőek, kiegészíthetőek, a szoftver jól strukturált lesz és átlátható. Az újabb funkciók implementációja során a megoldandó problémát is kisebb részekre tudjuk így bontani.

**Model:** Az MVC-ben az M a Model-t jelenti. Ebben a rétegben írjuk le az adatbázisban található entitásokat és ezek kapcsolatait.

**Controller:** A Controller feladata a beérkező HTTP kéréseket feldolgozni, és ezekre válaszolni. Végpontokat definiálunk, amelyeket a külvilág (Angular) alkalmazás elér HTTP kérésekkel és ezekre választ kap. Gyakran készítünk egy Service réteget is a Controller mellé, hogy még jobban szétválasszuk a felelősségi köröket: A controller csak a kérések fogadásával és a válaszadással foglalkozik, a Service réteg a kérés és válasz közötti adat feldolgozással, adatbázissal való kommunikációval.

**Service:** A Service réteg szigorúbban véve a Controller-hez tartozik, az üzleti logikát tartalmazza. A mi feladatunkban például a felhasználók validálásának menetét és a regisztrálást fogja tartalmazni.

**Data Access Layer:** Lényegében a Model-be tartozik, de csak most jutottunk el a használatához, ezért mutatom itt be.

**View:** A View rétegbe kerülnek azok a fájlok, amelyek a megjelenést biztosítják, tehát esetünkben a html fájlok, amelyekbe a böngésző értelmezni fog.

## Végpont-tervek

### Leírások

Jelen állapotban 3 db végpontunk van tesztelés céljából, de a későbbiekben bővíteni fog.

- Register
- Login
- Greet

### 1 db végpont működésének leírása(szekvenciadiagram)

