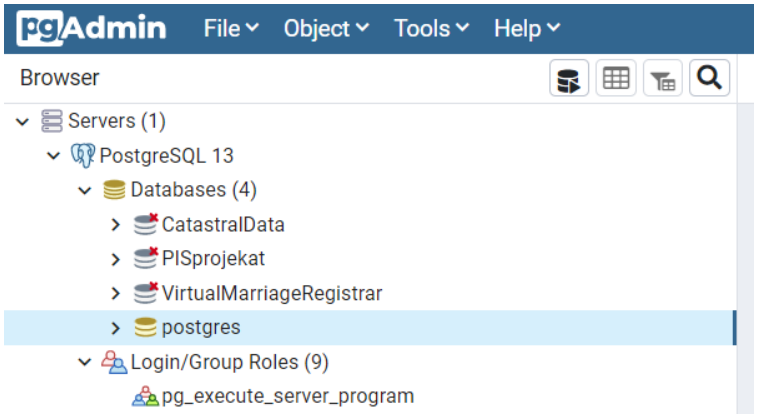
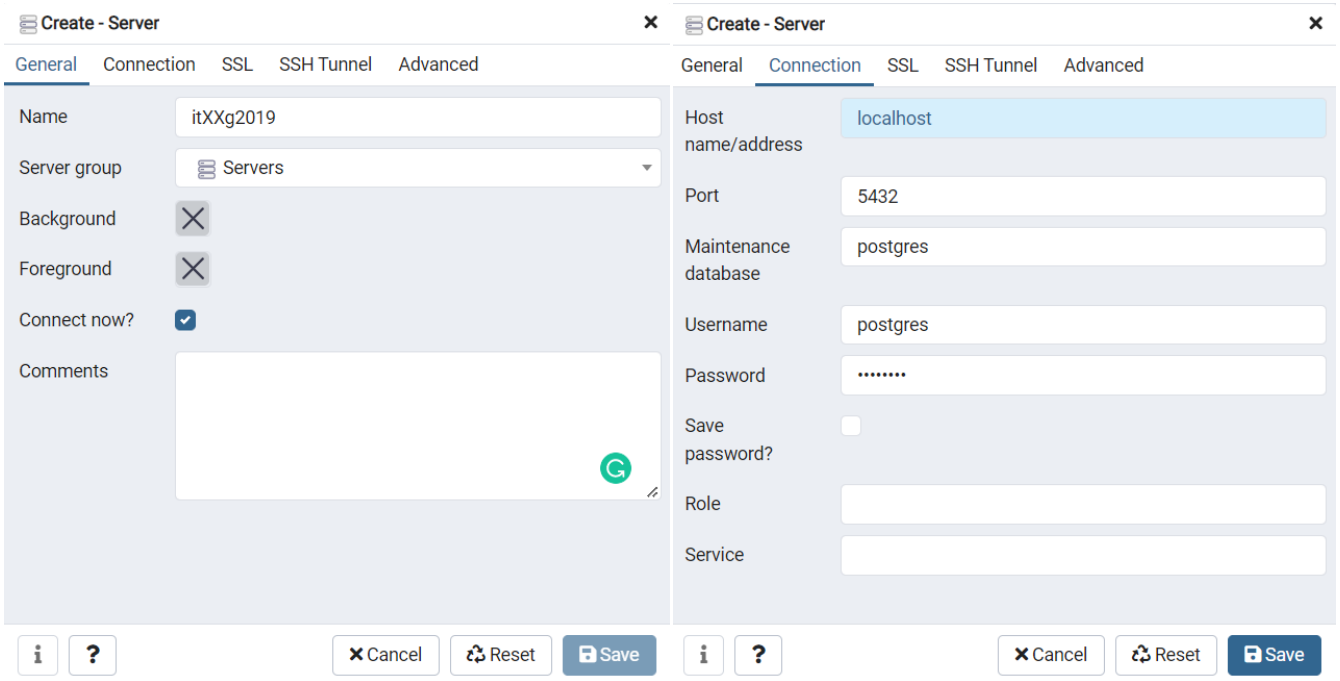
Preduslovi za rad:

- instaliran *PostgreSQL* (i *pgAdmin*),
- kreirane tabele u bazi podataka (pokrenut *CreateScript.sql* i *InsertScript.sql*) – skript fajlovi dostupni su na Sova platformi i
- instaliran *Spring Tool Suite.*

Kada **sa vaših ličnih računara** pokrenete *pgAdmin, u*glavnom konekciju na server nije potrebno kreirati već je ona automatski definisana. Na slici vidimo server pod nazivom *PostgreSQL* i u okviru njega *postgres* bazu podataka koju ćemo koristiti.



Ukoliko ipak nemate kreiran server, možete ga kreirati na sledeći način:



S obzirom na to da je ovo uputstvo pisano kako biste od kuće mogli da podesite sve što je potrebno, u polje *Host name* umesto IP adrese fakultetskog server unosite localhost.

Kada je sve podešeno, možete pokrenuti skript fajlove rađene na prethodnim vežbama.

Pokrenite *Spring Tool Suite* (STS). Za *workspace* postavite putanju do foldera u kom želite da čuvate projekte.

Za potrebe izrade projekta iz predmeta Razvoj višeslojnih aplikacija, nije potrebno kreirati projekat lokalno na računaru, već se projekat preuzima iz *GitHub* repozitorijuma koji je kreiran za svakog studenta.

Ulogujte se na *GitHub* - https://github.com/.

Da biste koristili *GitHub* potrebno je da kreirate *Personal Access Token* (PAT).

Uputstvo za kreiranje PAT možete pronaći na:

https://docs.github.com/en/github/authenticating-to-github/keeping-your-account-and-data-secure/creating-a-personal-access-token

Kreirani token sačuvajte u txt fajlu kako biste kasnije mogli da ga kopirate i koristite.

Ukoliko prvi put koristite token u okviru STS okruženja, potrebno je da odete na:

1) **Window** > **Preferences** > **General** > **Security** > **Secure Storage**

2) i da odčekirate **UI Prompt**

Potrebno je prijaviti se na *GitHub Classroom* na adresi https://classroom.github.com/a/6hiza1qX i izabrati svoje ime, prezime i broj indeksa.

Nakon prijave, svakom studentu će biti dodeljen poseban repozitorijum koji je potrebno preimenovati. Preimenovanje se vrši klikom na Settings, a potom unosom novog imena. Novo ime treba da bude u formatu **brojIndeksa-godinaUpisa-Prezime-Ime** (npr. IT01-2017- Petrovic-Petar).

Adresu repozitorijuma iskoristiti za preuzimanje „praznog“ (inicijalnog) projekta. Kopirati putanju do vašeg repozitorijuma.

Povezivanje se vrši u okviru STS-a na sledeći način:

STS: **File** -> **Import**... -> **Git** -> **Project from Git** -> **Clone URI** (nalepi se URI od repozitorijuma na *GitHub*-u)

Unosite korisničko ime i PAT na mestu lozinke.

Preuzimanje inicijalnog projekta je završeno, pokrenite aplikaciju kako biste se uverili da se aplikacija uspešno pokreće.

Ulazna tačka u projekat je MAIN metoda koja se nalazi unutar klase *src/main/java/rva/**BackendRvaApplication.java***.

Napomena: Nemojte se zbuniti, paket koji se u mom projektu zove “rva” kod vas je “rppstart”.

Anotacije se koriste kako bismo u određene klase, metode, obeležja injektovali neke konfiguracije. Dodaju se na postojeći Java kod.

@SpringBootApplication – uvek se postavlja za klasu koja je zadužena za pokretanje projekta i ta klasa mora da se nalazi u root-nom paketu (rva).

src/main/resources/**application.properties**

- Konfiguracije koje se tiču celog projekta.

Prekopisatati sledeći kod u navedeni fajl.

```
spring.jpa.database = POSTGRESQL
spring.datasource.platform = postgres
spring.jpa.show-sql = true
spring.datasource.driver-class-name = org.postgresql.Driver

# parametri za konektovanje na bazu podataka sa fakultetskih računara
# za studente je username i naziv baze u formatu: itXXg20XX, a password je: ftn

#spring.datasource.url = jdbc:postgresql://192.168.100.251:5432/postgres
#spring.datasource.username = postgres
#spring.datasource.password = postgres

# Parametri za konektovanje na bazu podataka sa ličnih računara
# Svima je username i password postgres!
spring.datasource.url = jdbc:postgresql://localhost:5432/postgres
spring.datasource.username = postgres
spring.datasource.password = postgres

# port pomocu kojeg pristupamo app
server.port = 8083

# disable driver's feature detection
spring.jpa.properties.hibernate.temp.use_jdbc_metadata_defaults = false

# without detection you have to set the dialect by hand
spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.PostgreSQLDialect
```

Prilikom kreiranja projekta dodali smo *Spring Web* čime smo preuzeli i *Tomcat web server*, na kom se pokreće naša backend aplikacija.

Pom.xml – Opis SpringBoot projekta.

Kreiramo novi paket unutar već postojećeg rva paketa (desni klik na več postojeći rva paket -> New -> Package). U okviru ovog paketa biće klase kontrolera koje ćemo koristiti.

src/main/java/rva

rva.**ctrls**

Unutar tog paketa kreiramo novu klasu **HelloRestContorller**.

Dodajemo anotaciju **@RestContorller**. Ova anotacija koristi se samo na nivou klase i koristi se za definisanje RESTful veb servisa. Sadrži osobine sledeće dve anotacije:

1. **@Controller** – specificira klasu koja će se ponašati kao kontroler i
2. **@ResponseBod**y – rezultat izvršavanja će se smestiti u response body (koji se isporučuje u JSON formatu).

Unutar ove klase kontrolera dodajemo metodu. Anotacija **@RequestMapping** mapira veb zahteve na određene metode. Podrazumevani zahtev je GET. Pokrenuti i u pretaživaču učitati adresu http://localhost:8083/ .

```
@RequestMapping("/")
public String helloWorld() {

	return "Hello World!";

}
```

Dodajemo još jednu metodu koja će računati zbir dva random broja. Sačuvati i učitati adresu http://localhost:8083/zbir.

```
@RequestMapping("zbir")
public String zbir() {
	long x = Math.round(Math.random()*10);
	long y = Math.round(Math.random()*10);
	return x + " + " + y + " = " + (x+y);
}
```

U okviru klase *src/main/java/rva/**BackendRvaApplication.java*** dodajemo još jednu metodu koja će sve Bean-ove ispisati na konzoli da bismo videli šta sve u pozadini Spring pravi za nas.

Šta je Bean? Objekti koji formiraju aplikaciju, a koji se nalaze u nadležnosti Spring kontejnera zovu se zrna (beans).

@Bean anotacija govori Spring-u da u konfiguracionoj klasi metoda koja je anotirana tom anotacijom treba da bude registrovana kao bean.

```
@Bean
public CommandLineRunner commandLineRunner(ApplicationContext ctx) {

	return args -> {
		System.out.println("Beans provided by Spring Boot: ");
		String[] beanNames = ctx.getBeanDefinitionNames();
		Arrays.sort(beanNames);
		for(String beanName: beanNames) {
			System.out.println(beanName);
		}
	};


}
```

Kada smo završili sa izmenama na projektu, želimo da ih postavimo u *GitHub* repozitorijum. Pratimo sledeće korake:

STS: Window -> Show view -> Other.. -> Git -> Git Staging

U okviru STS okruženja pojavljuje se prozor u kom možemo pratiti fajlove koji su izmenjeni a još uvek nisu postavljeni u povezani *GitHub* repozitorijum.

Sve iz odeljka *Unstaged Changes* prevučemo u odeljak *Staged Changes*. Dodamo odgovarajući *Commit Message* i zatim možemo da uradimo *Commit and Push*.

Kompletan kod sa vežbi možete pogledati u okviru GitHub repozitorijuma zajedničkog za celu grupu:

https://github.com/Razvoj-viseslojnih-aplikacija-2021-2022?q=&type=public&language=&sort=

Napomena:

Ako ne možete da koristite određene simbole na tastaturi (najčešće **{**) znači da je određena kombinacija tastera sačuvana kao prečica za neku komandu. Kako bi se to rešilo, potrebno je uraditi sledeće:

Window -> Preferences (u search ukucate **Keys**) ->

Nađete komandu za tu kombinaciju tastera (za { je Alt + B) i -> Unbind Command