

# Drone Services App

Eugeniu Orlov



# Agenda

---

Descrierea aplicației

---

Criterii de design

---

Arhitectura generală a aplicației

---

Backend

---

Frontend

---

Structura bazei de date

---

Securitatea aplicației





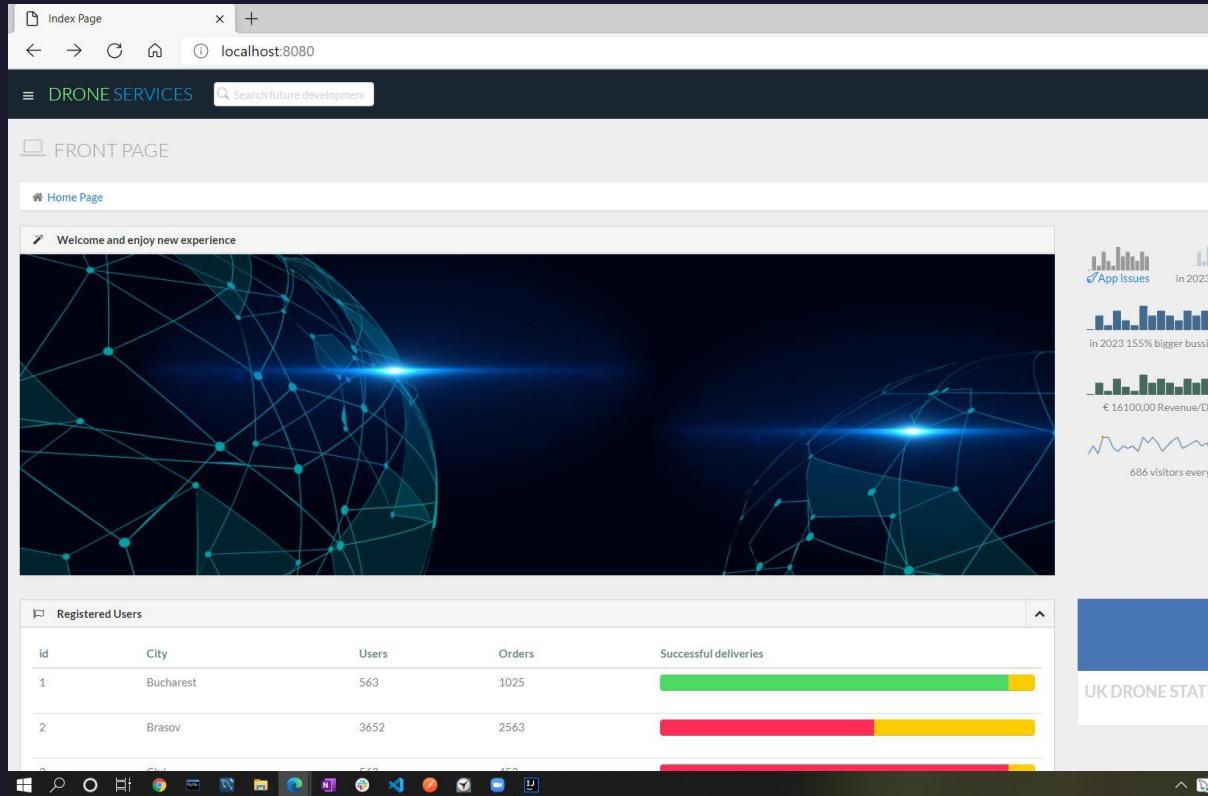
# Introducere

- APlicație web pentru servicii cu drone
- Aplicația permite utilizatorilor să acceseze servicii precum livrarea de pachete mici sau servicii de securitate la distanță și transport persoane / animale, în zone greu accesibile, cu ajutorul dronelor



# Descrierea aplicației

# Descrierea aplicației



- Aplicația modulară integrează servicii cu drone, având capacitatea să calculeze distanțele, costurile și capacitatea de transport în funcție de greutate și de gradul de încărcare al bateriilor.
- Conceptul inițial se bazează pe livrări de pachete mici din punct fix în punct fix cu autonomie de zbor până la 5km în aceeași localitate, respectiv cu verificarea disponibilității serviciului în localitatea indicată de client.
- Aplicația generează statistici în caz de indisponibilitate și asigură analizarea creării unor noi centre geografice în funcție de cerere.

# Descrierea aplicației



## SISTEMELE PRINCIPALE DE FUNCȚIONARE

Localitatea

- Regiunea de acțiune
- Managementul listei de locații

Clienții

- Managementul Listei de clienți

Dronele

- Tipuri de drone
- Managementul Listei de drone



## TEHNOLOGII UTILIZATE

Java

JavaScript

Spring Security - user login

Spring Data JPA

Hibernate

MySQL

Thymeleaf

HTML

CSS

Bootstrap și Jquery

GitHub



## ENTITĂȚILE DE BAZĂ

(geografic) - Oras/localitate

Nume, Descriere, Cod postal/cordonate geografice

(users) - Client

Nume, Adresa, Orasul/localitatea, E-mail, Telefon

(services) - Drone

Nume, disponibilitate

(services) - Tipul dronei

Descriere, Detalii

(geografic) - Locatii disponibile pentru preluari/livrari indirekte

Adresa, Descriere, Detalii

(geografic) - Zone de actiune/regiuni

Nume comun/cartier, Detalii, Coordonate/cod postal

# Aplicația este împărtită în:



## B A C K E N D

- Partea funcțională a aplicației care face legătura cu baza de date și interacționează exclusiv cu programatorul.

## F R O N T E N D

- Partea aplicației web vizibilă în interacțiunea cu utilizatorul, compusă din:
  - Zona dedicată administratorului bazei de date
  - Zona dedicată tuturor vizitorilor site-ului



# Criterii de design



# Criterii de design

Sistemul creat este destinat companiilor de curierat, instituțiilor sau companiilor care transmit plicuri/pachete mici în mod curent între locații fixe.

Ca atare, aplicația trebuie să poată fi **personalizată ușor** și să ofere funcționalități de **management de bază** pentru fiecare entitate (*creare, editare, stergere*).



# Funcționalități de bază

Locațiile	Dronele	Clienții	Plata	Statistici & rapoarte
<ul style="list-style-type: none"><li>• Editabile în funcție de cerere</li><li>• Clasate pe regiuni de acțiune</li><li>• Interconectate cu cererea clienților</li></ul>	<ul style="list-style-type: none"><li>• Alocate între 2 locații din raza de acțiune / regiune</li><li>• Utilizabile de mai mulți clienți</li><li>• Livrare imediată sau cu temporizator</li></ul>	<ul style="list-style-type: none"><li>• Predefiniți sau înrolați extern</li></ul>	<ul style="list-style-type: none"><li>• Abonament pentru disponibilitatea serviciilor &amp;</li><li>• Plata raportat la utilizarea efectivă a serviciilor<ul style="list-style-type: none"><li>• Taxa pe pachet (greutate, distanță)</li></ul></li></ul>	<ul style="list-style-type: none"><li>• Grad de încărcare per locație</li><li>• Grad de încărcare per dronă</li><li>• Reușita de reîncărcare completă a dronelor per intervale de timp</li><li>• Reușita de livrare în funcție de condițiile exterioare de zbor</li><li>• Întârzieri la livrare</li><li>• Eșuarea livrării</li></ul>

# Constrângeri de business

Clarificarea cadrului legal aplicabil privind utilizarea spațiului aerian

Identificarea clienților potențiali

Identificarea și atragerea investitorilor

Identificarea de parteneri și furnizori

Pentru echipamente UVA

Pentru stații de reîncărcare și soluții de automatizare

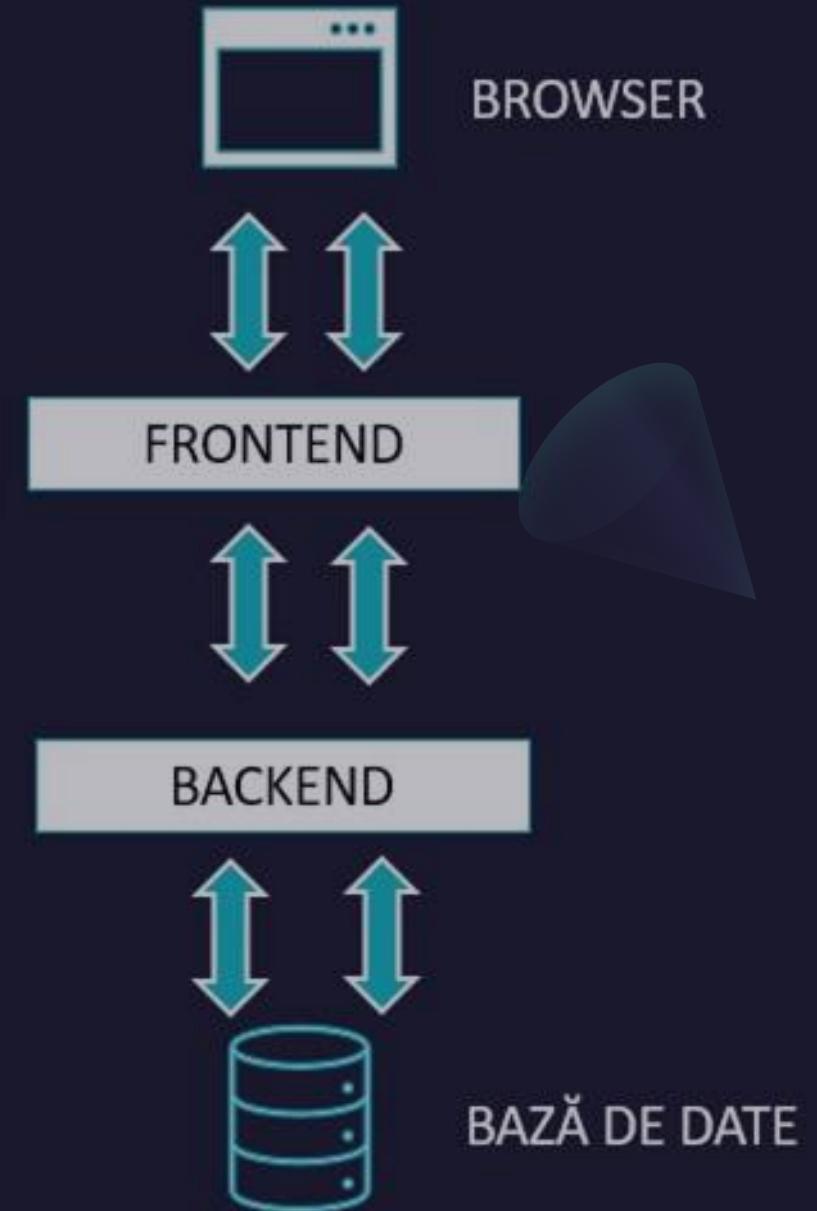
Pentru platformele de preluare și livrare a pachetelor



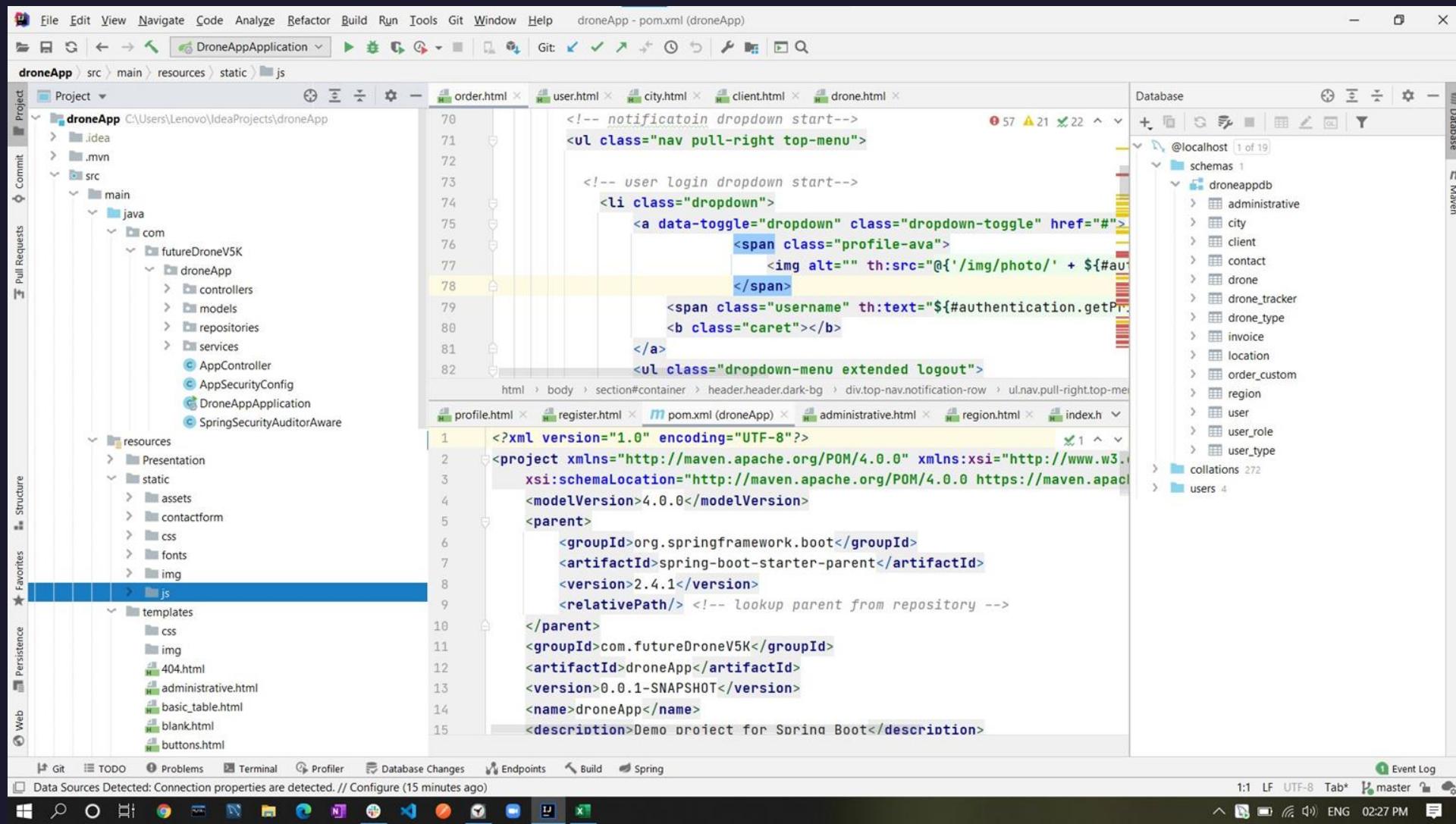
# Arhitectura generală a aplicației



# Arhitectura generală a aplicației



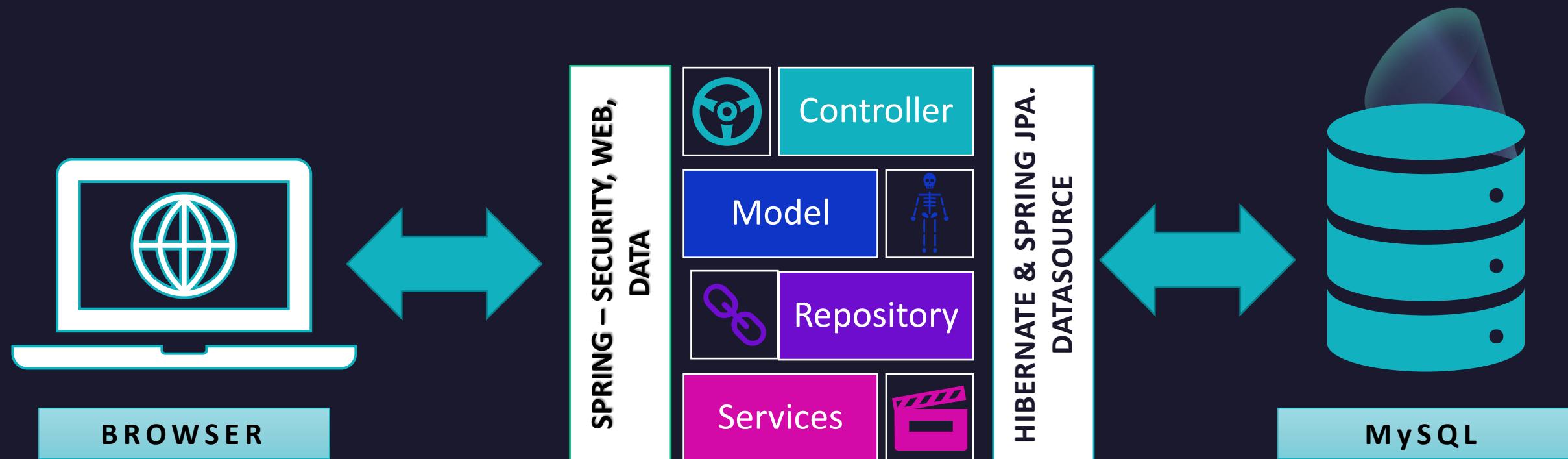
# Structura proiectului



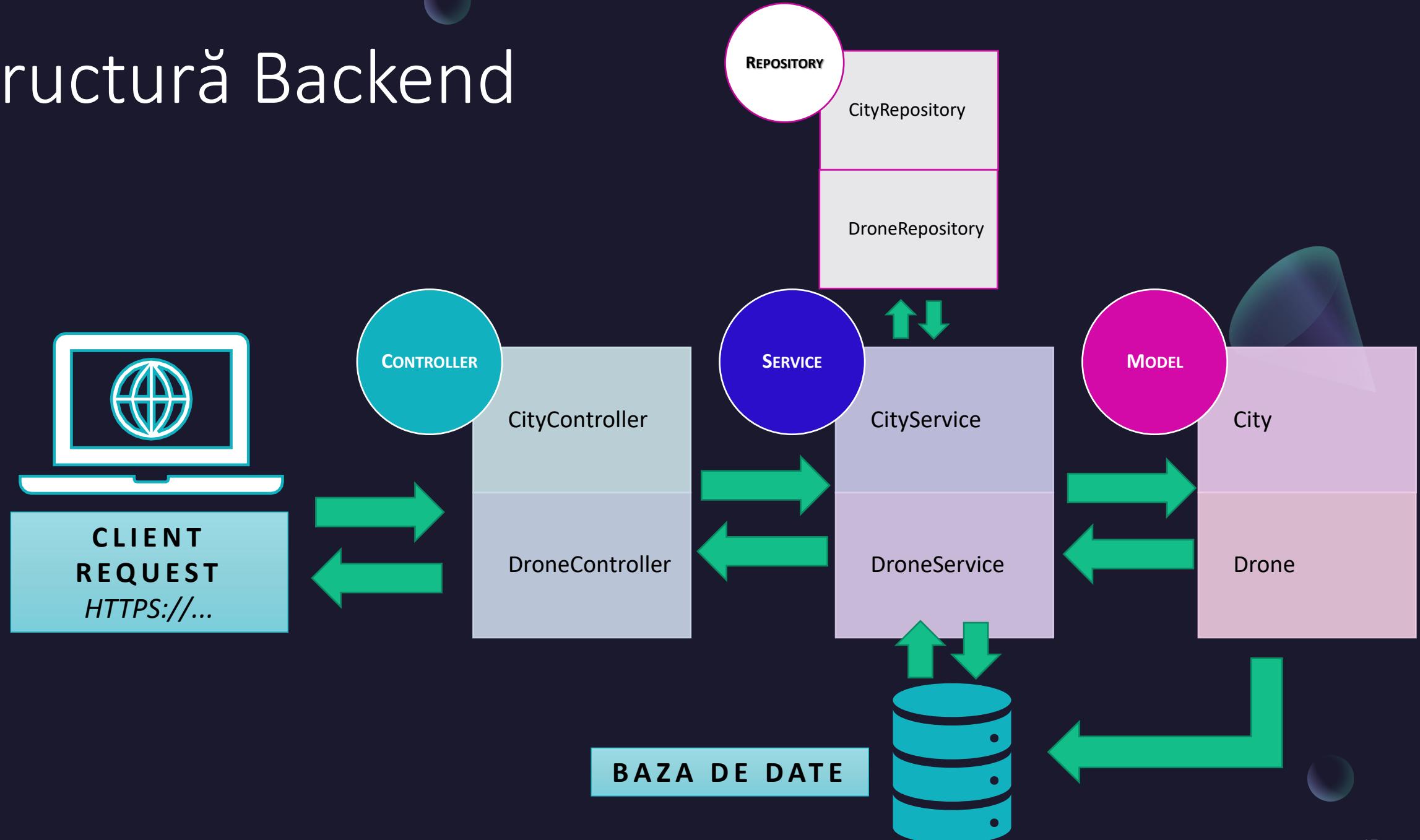


Backend

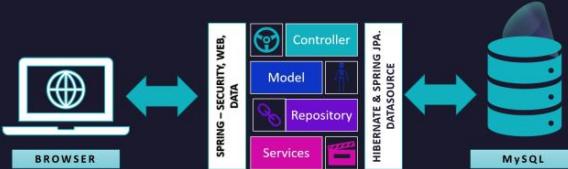
# Arhitectură Backend



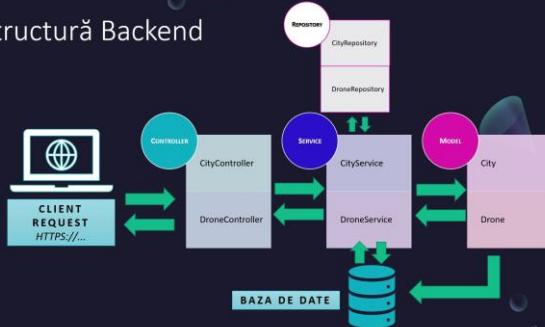
# Structură Backend



Arhitectură Backend



Structură Backend



The screenshot shows an IntelliJ IDEA interface with several code editor panes open, displaying Java code for a Spring Boot application named "droneApp".

- City.java:** A JPA entity class with annotations like @Entity, @Data, and @Id.
- CityController.java:** A controller class with methods for GET /cities, POST /cities/addNew, PUT /cities/update, and DELETE /cities/delete.
- CityService.java:** A service class with methods for getting cities, saving cities, finding cities by ID, and deleting cities.
- CityRepository.java:** A repository interface extending JpaRepository.

The code uses Lombok annotations (@Data, @GeneratedValue) and Spring annotations (@Entity, @Autowired, @GetMapping, etc.). The project navigation bar at the top shows "droneApp - CityController.java". The bottom status bar indicates "30:36 CRLF UTF-8 4 spaces master ENG 03:59 PM".

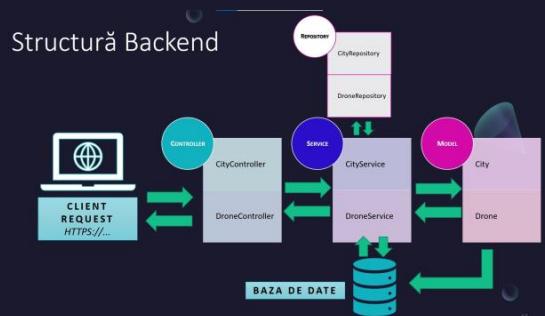
The screenshot shows an IDE interface with multiple tabs open, displaying Java code for a Spring Boot application.

**Project Tree:** On the left, there's a tree view showing the project structure: `droneApp` > `src` > `main` > `java` > `com.futureDroneV5K.droneApp.controllers` > `DroneController`.

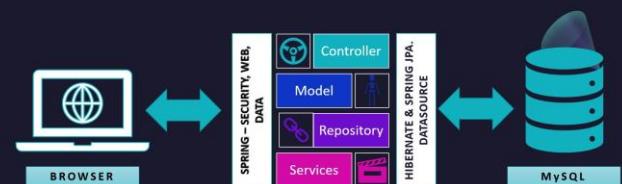
**Code Editors:**

- DroneService.java:** Contains methods for interacting with a `DroneRepository`. It includes `@Service`, `@Autowired`, and various `get`, `save`, `delete`, and `findById` methods.
- Drone.java:** A POJO with annotations like `@AllArgsConstructor`, `@JsonIdentityInfo`, `@Id`, `@GeneratedValue(strategy = GenerationType.IDENTITY)`, `@Column(name = "droneid")`, and `@OneToMany` relationships.
- DroneRepository.java:** An interface extending `JpaRepository<Drone, Long>`.
- DroneController.java:** A controller class annotated with `@Controller`. It contains `@Autowired` fields for `droneService`, `cityService`, and `regionService`. It defines REST endpoints using `@GetMapping`, `@PostMapping`, `@PutMapping`, `@DeleteMapping`, and `@PatchMapping`.

**Toolbars and Status Bar:** The top bar includes standard file operations (File, Edit, View, Navigate, Code, Analyze, Refactor, Build, Run, Tools, Git, Window, Help) and a tab for `droneApp - DroneController.java`. The bottom status bar shows the current time as 05:17 PM and other system information.

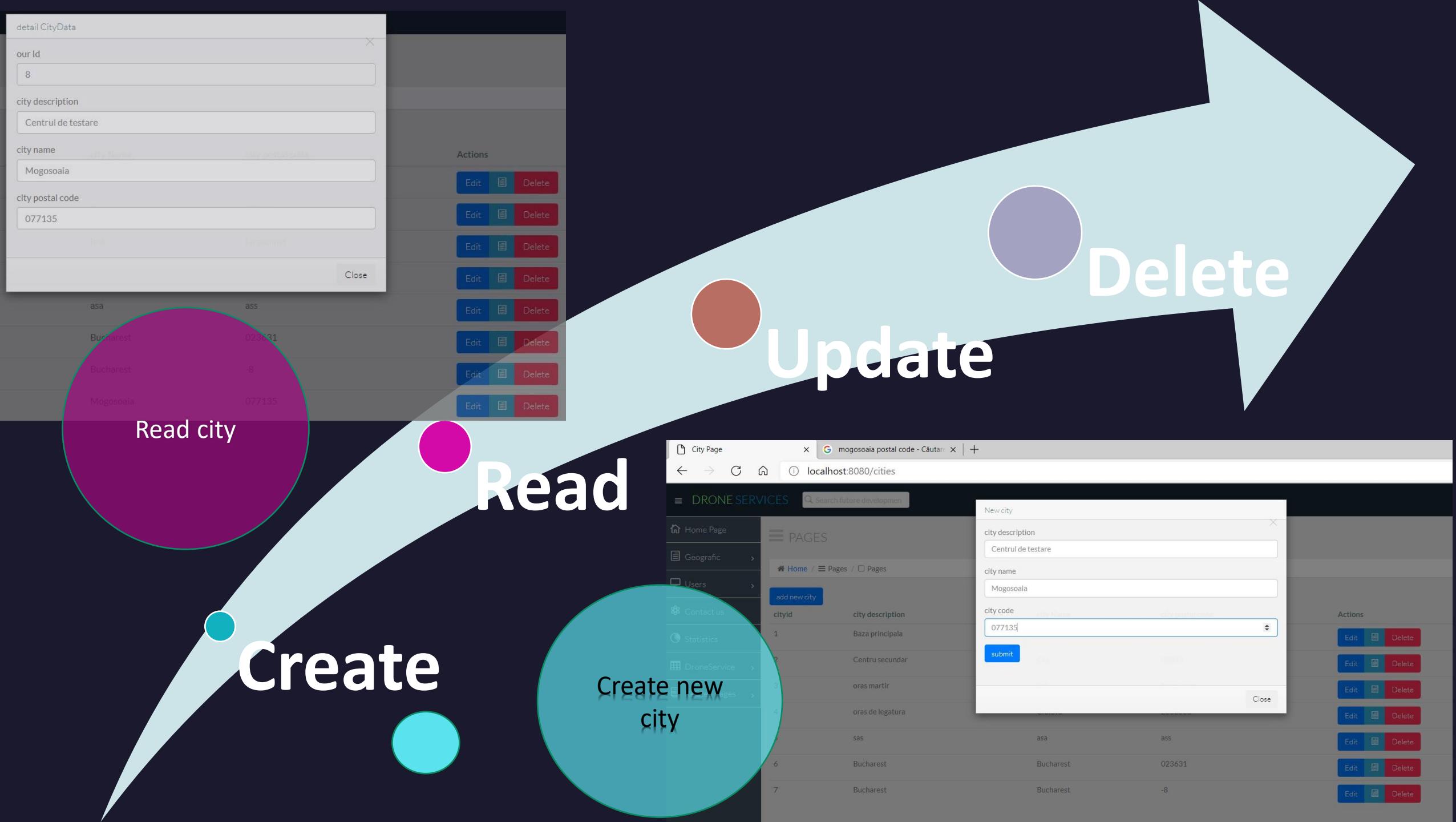


Arhitectură Backend



# Frontend





The image is a collage of screenshots illustrating CRUD operations on a city database, overlaid with large white text labels: 'Create', 'Read', 'Update', and 'Delete'.

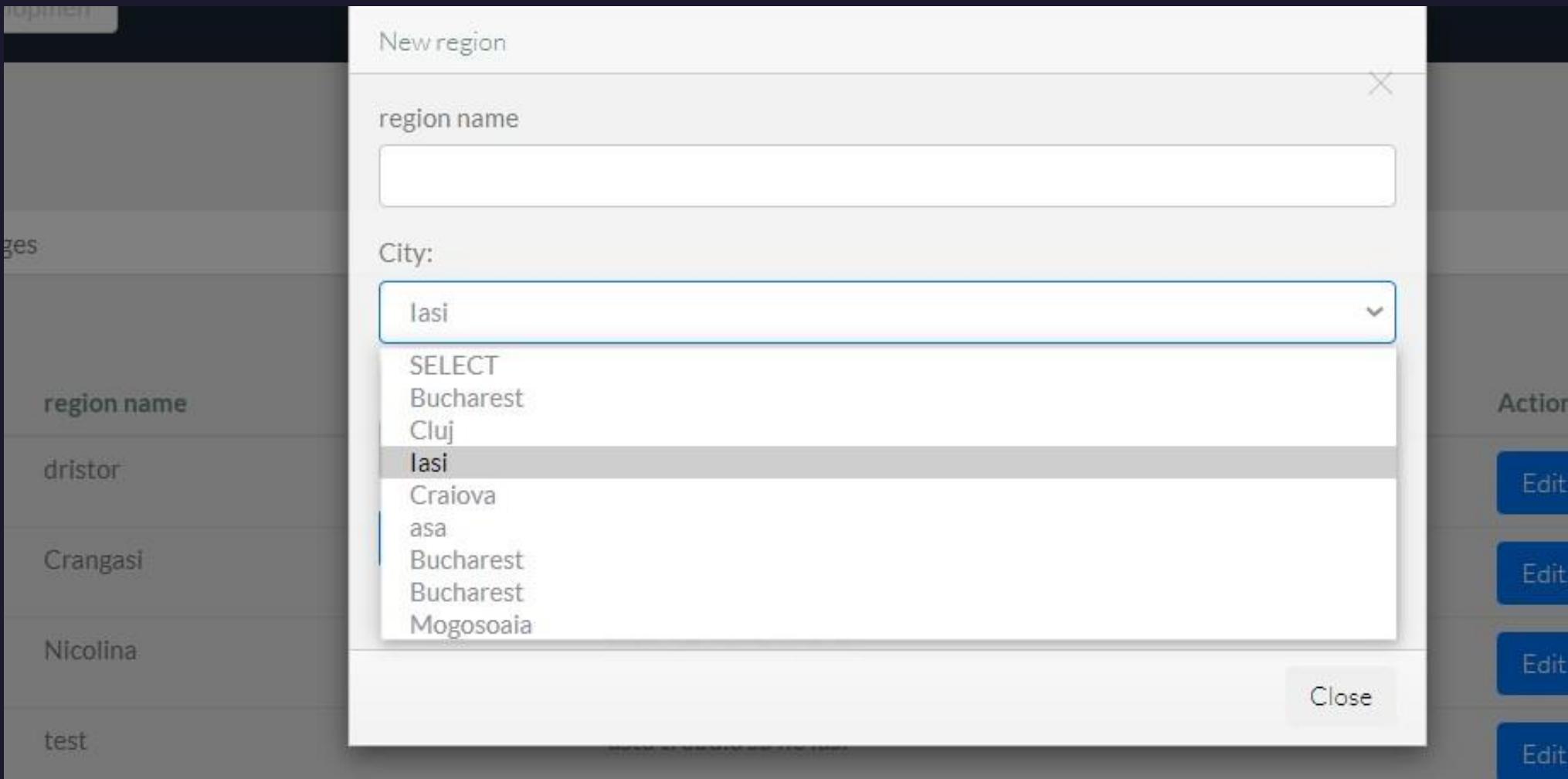
- Create:** A screenshot of a modal window titled "Update CityData". It contains fields for "city id" (8), "city description" ("Centrul de testare aerospaciala cu laser"), "city name" ("Mogosoaia"), and "city postal code" ("077135"). A "submit" button is at the bottom left, and a "Close" button is at the bottom right.
- Read:** A screenshot of a table showing city data. The columns are "cityid" and "city description". The rows show:
  - 1 Baza principala
  - 2 Centru secundar
  - 3 oras martir
  - 4 Oras de legatura
  - 5 sas
  - 6 Bucharest
  - 7 Bucharest
  - 8 Mogosoaia
- Update:** A screenshot of a table showing city data. The columns are "cityid", "city name", and "city postal code". The rows show:
  - 1 Bucharest 023631
  - 2 Bucharest -8
  - 3 Mogosoaia 077135A "Actions" column on the right contains "Edit" and "Delete" buttons for each row.
- Delete:** A screenshot of a modal window titled "confirm delete". It asks "Sunteti sigur ca doriti sa stergeti orasul selectat?". It has "Confirm - delete" and "Cancel action" buttons.
- Code View:** A screenshot of a browser developer tools' element inspector. It shows the HTML structure of the "confirm delete" modal and the JavaScript code for handling the delete button click.

Large white text labels are overlaid on the image:

- 'Create' is positioned near the 'Create' screenshot.
- 'Read' is positioned near the 'Read' screenshot.
- 'Update' is positioned near the 'Update' screenshot.
- 'Delete' is positioned near the 'Delete' screenshot.

# User Interface

PICK CITY FROM EXISTING LIST



# User Interface

CREATE NEW LOCATION BY  
SELECTING EXISTING CITY & REGION

The screenshot shows a development environment with the following components:

- Code Editor:** Displays `location.js` with code for handling location creation. It includes a dropdown menu for selecting cities and regions, and input fields for name, description, and details.
- Browser Preview:** Shows a modal dialog titled "New Location" with fields for City (Bucharest), Region (Crangasi), name/description of location (Piata), and other details (intrarea dinspre Lid). A "submit" button is at the bottom.
- Database Explorer:** Shows the database structure for the `droneappdb` schema, including tables like `city`, `region`, and `location`.
- Bottom Bar:** Includes icons for Git, Run, TODO, Problems, Terminal, Profiler, Database Changes, Endpoints, Build, and Spring.

# Baza de date



# Baza de date MySQL

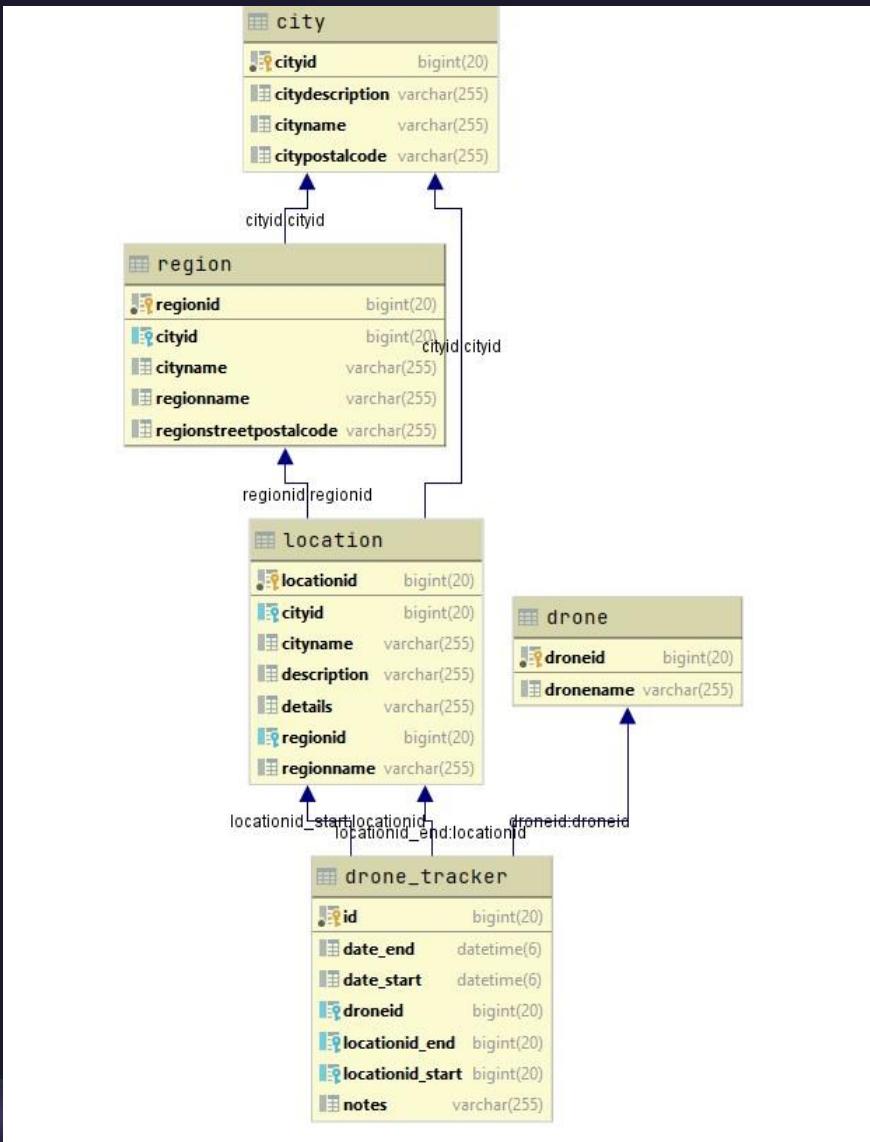


DIAGRAMA REGION/DRONE

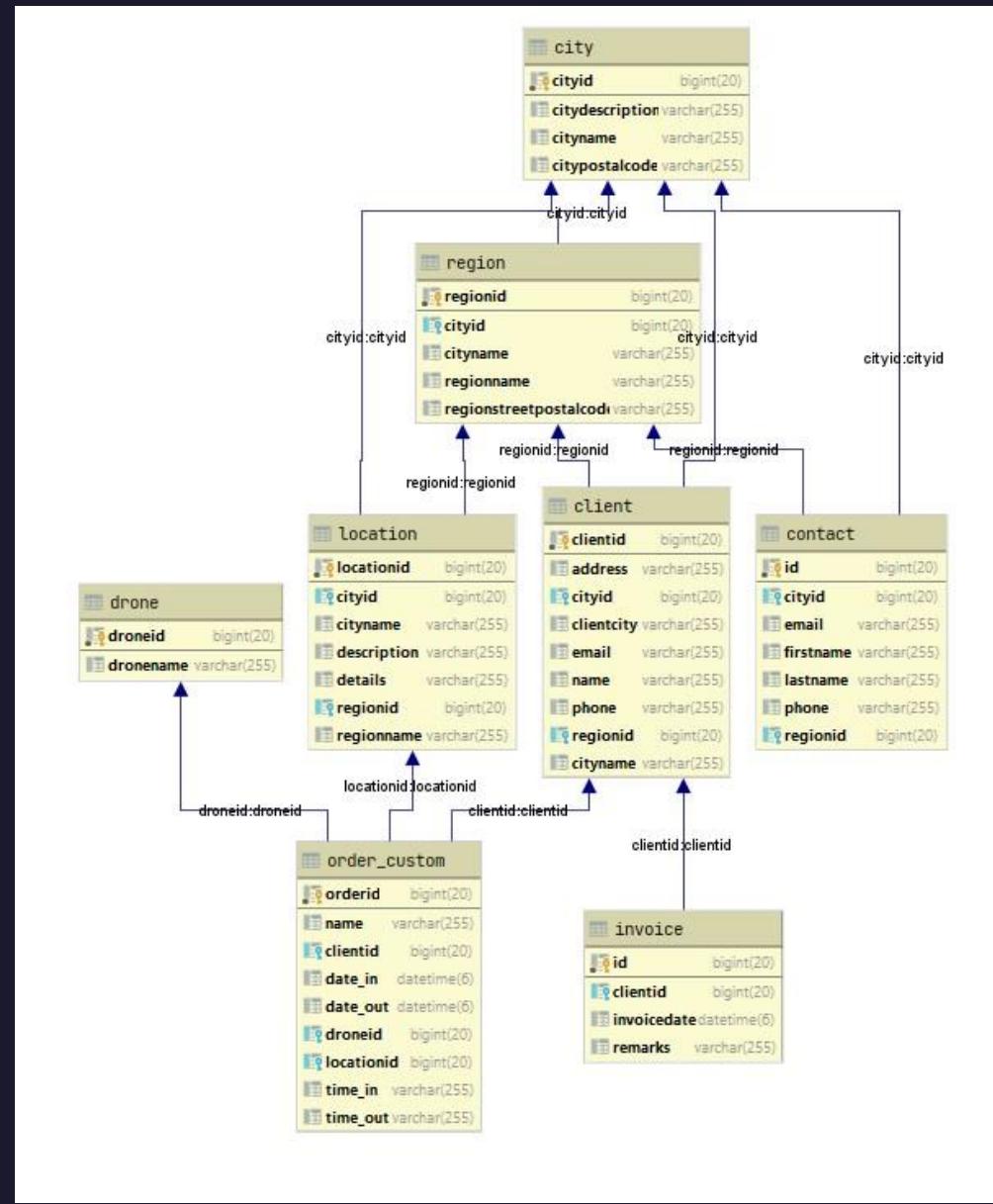
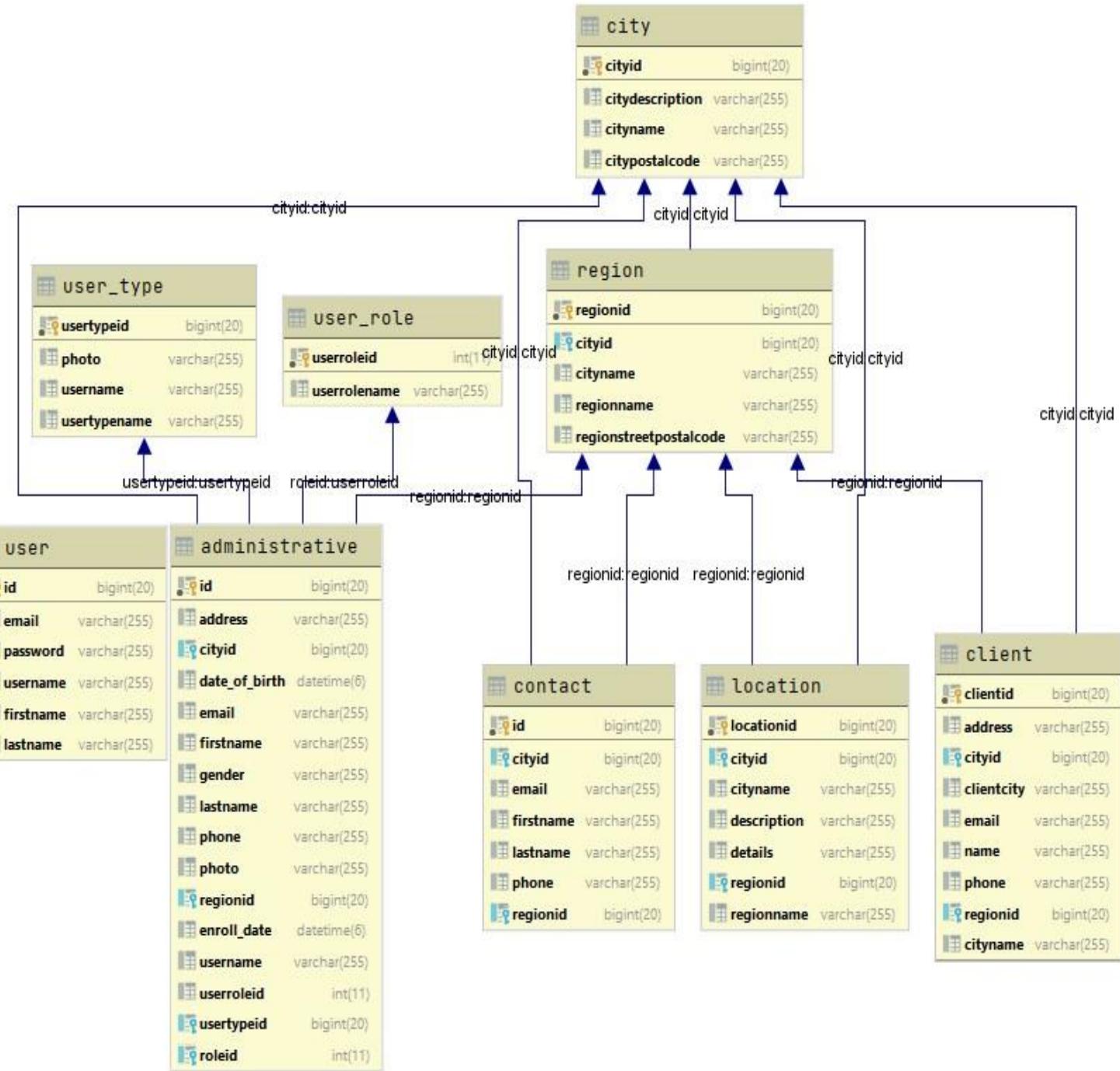


DIAGRAMA ORDER BY REGION

# Baza de date MySQL

DIAGRAMA USER BY REGION



# Elemente de securitate



The screenshot shows an IDE interface with multiple tabs open, displaying Java code for a Spring Boot application named "droneApp".

**Project Tree:**

- droneApp
- src/main/java/com/futureDroneV5K/droneApp
- AppSecurityConfig.java
- DroneAppApplication.java
- AppController.java

**Code Editor (Top Right):** AppSecurityConfig.java

```
17 @EnableWebSecurity
18 @Configuration
19 public class AppSecurityConfig extends WebSecurityConfigurerAdapter {
20     @Override
21     protected void configure(HttpSecurity http) throws Exception {
22         http
23             .csrf().disable()
24             .authorizeRequests()
25                 .antMatchers("/login", "/resources/**", "/css/**")
26                 .antMatchers("/register", "/resources/**", "/css/**")
27                 .antMatchers("/users/addNew").permitAll()
28             .anyRequest().authenticated()
29             .and()
30             .formLogin()
31                 .loginPage("/login").permitAll()
32             .and()
33             .httpBasic()
34                 .logout()
35                     .invalidateHttpSession(true)
36                     .clearAuthentication(true)
37             .logoutRequestMatcher(new AntPathRequestMatcher("/logout"))
38             .logoutSuccessUrl("/login").permitAll();
39     }
40     @Bean
41     public PasswordEncoder passwordEncoder() { return new BCryptPasswordEncoder(); }
42     @Bean
43     public BCryptPasswordEncoder bCryptPasswordEncoder() { return new BCryptPasswordEncoder(); }
44     @Autowired
45     private UserDetailsService userDetailsService;
46     // DAO
47     @Bean
48     public AuthenticationProvider authenticationProvider() {
49         DaoAuthenticationProvider provider = new DaoAuthenticationProvider();
50         provider.setUserDetailsService(userDetailsService);
51         provider.setPasswordEncoder(passwordEncoder());
52         return provider;
53     }
54 }
```

**Code Editor (Bottom Left):** SpringSecurityAuditorAware.java

```
8
9     public class SpringSecurityAuditorAware implements AuditorAware<String> {
10
11         @Override
12         public Optional<String> getCurrentAuditor() {
13             // sec context holder
14             Authentication authentication = SecurityContextHolder.getContext().getAuthentication();
15             String username = authentication.getName();
16             return Optional.ofNullable(username).filter(s -> !s.isEmpty());
17         }
18     }
19 }
```

**Toolbars and Status Bar:**

- File Edit View Navigate Code Analyze Refactor Build Run Tools Git Window Help
- droneApp - AppSecurityConfig.java
- Git: ✓ ✓ ✓ ✓
- Project Commit Pull Requests Favorites Persistence Web
- Git Run TODO Terminal Profiler Database Changes Endpoints Build Spring
- Build completed successfully with 2 warnings in 4 sec, 664 ms (3 minutes ago)
- 45:6 CRLF UTF-8 4 spaces master 05:50 PM



SIGN UP



Username

Password

Remember me

Forgot Password?

Login

Signup



REGISTER



loan

Popescu

ipopescu@jmail.com

ipopescu

\*\*\*\*\*

\*\*\*\*\*

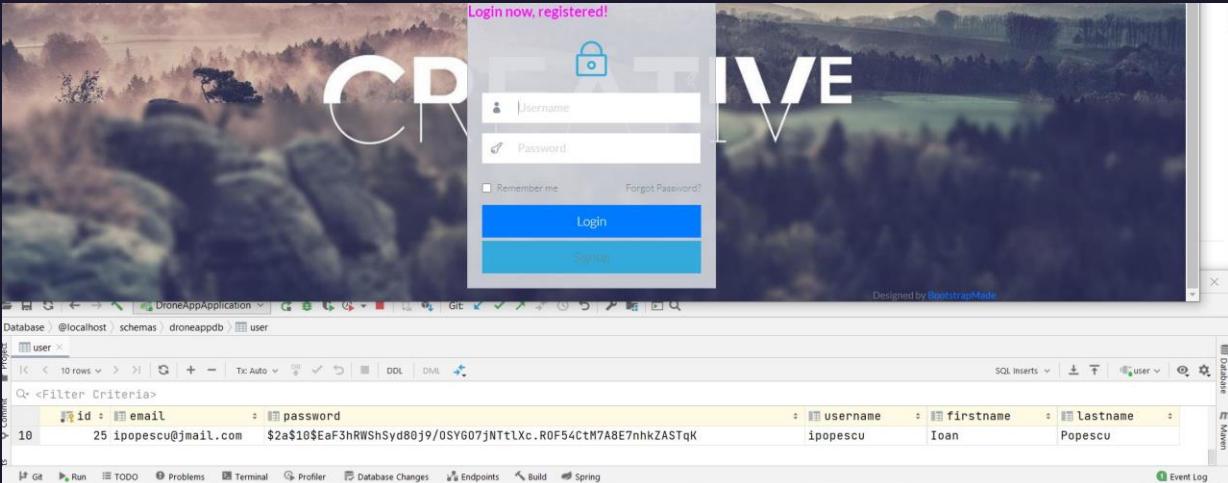
login page

Forgot Password?

Register



REGISTRATION COMPLETE



Login now, registered!

CD LIVE

Username: ipopescu

Password: \$2a\$10\$EaF3hRWSyhd80j9/OSYGO7jNTtLxc.R0F54CtM7A8E7nhkZASTqK

firstname: Ioan

lastname: Popescu

SQL Inserts

ID	Email	Password	Username	Firstname	Lastname
10	ipopescu@jmail.com	\$2a\$10\$EaF3hRWSyhd80j9/OSYGO7jNTtLxc.R0F54CtM7A8E7nhkZASTqK	ipopescu	Ioan	Popescu



# Multumesc!

Eugeniu Orlov

