Технически университет - София

Разработване на уеб приложение за резервация на спортни съоръжения

Дипломна работа

На

Изабел Христова Филипова

Цел и изходна постановка на дипломната работа

- Цел: Да се разработи уеб приложение за резервация на спортни съоръжения, използващо зададените в изходната постановка технологии

- Изходна постановка:
  - Java
  - Spring Boot
  - Hibernate
  - MySQL

Разгледани софтуерни продукти

- „Tereni.bg“
- „easybook.bg“
- „Sport4All.bg“
- Онлайн система за резервация на автобусни билети
- Онлайн система за болнични резервации
- Мобилно приложение за резервация на автобусни билети
- Приложение за онлайн резервации на незаети билети за влакове

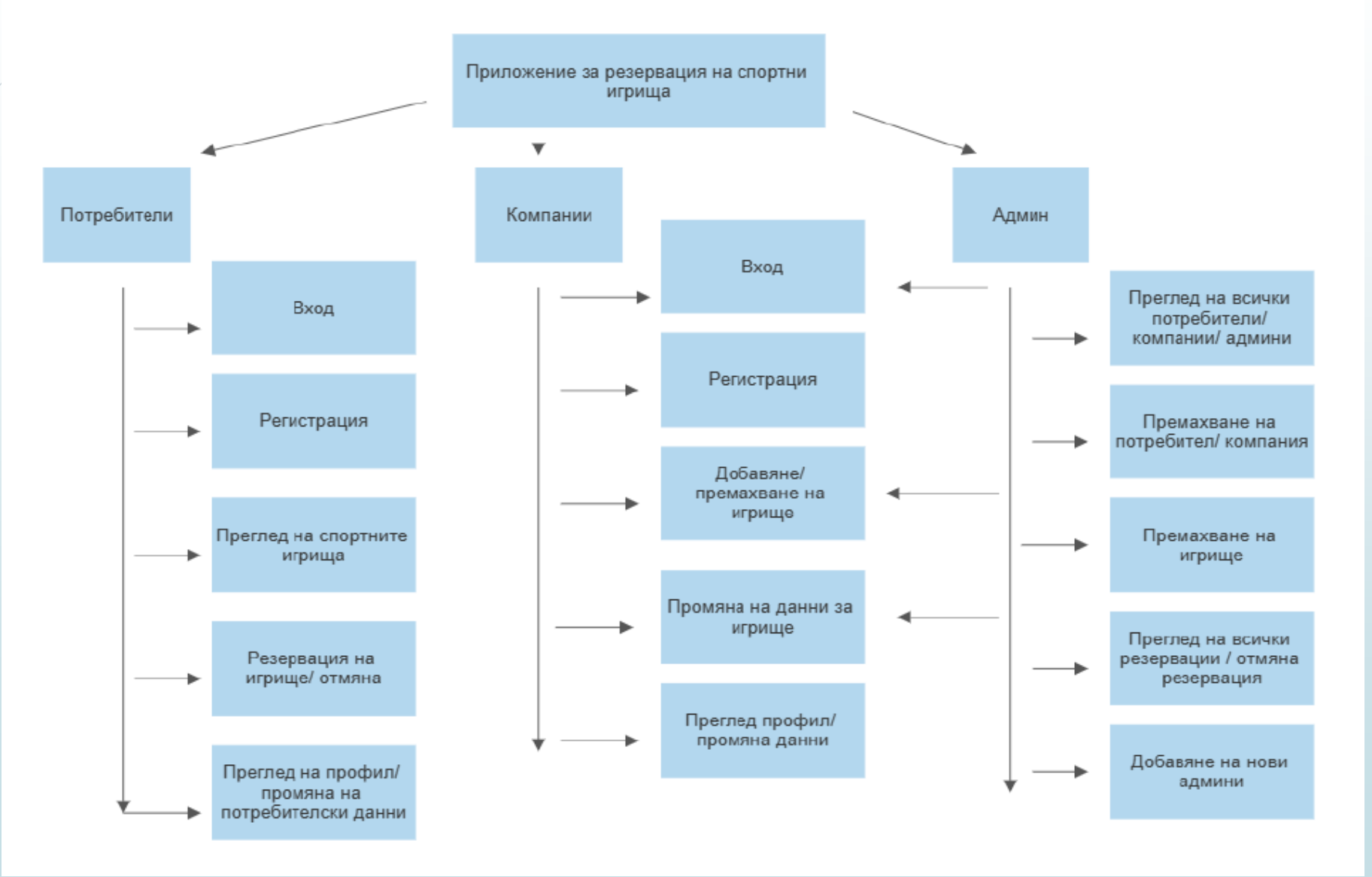
Недостатъци на разгледаните платформи

- Резервация, обвързана с непосредствено плащане
- Ограничен обхват на приложението само за даден град
- Липса на регистрация на потребителите
- Невъзможност за отмяна на резервация
- Липса на история за извършените резервации
- Достъп до обектите за резервация само след вход/регистрация

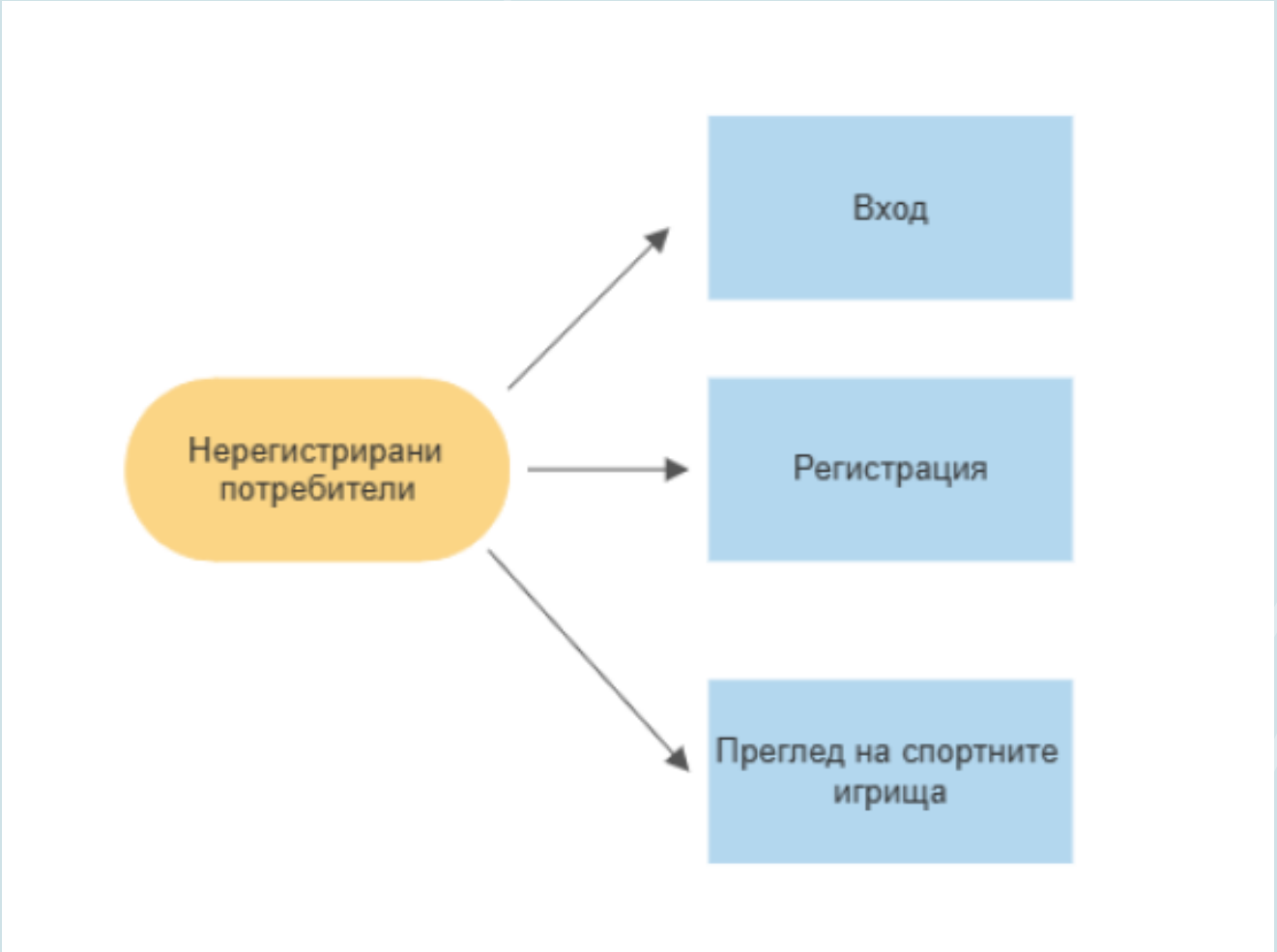
Подобрения при разработването дипломен проект

- Резервиране на игрище, без нужда от плащане
- Отмяна на резервация по всяко време
- Добавяне на спортни игрища от цялата страна
- Свободно разглеждане на всички игрища
- История на резервациите на всеки потребител

Архитектура на разрабтваното приложение



Операции, извършвани от анонимен потребител



Компоненти на програмната реализация

- Обекти – Entities
- Услуги – Services
- Управление – Controllers
- Хранилища – Repositories
- Сигурност и нейната конфигурация - SecurityConfig

Обекти - Entities

Обект „Потребители“ – Users Entity

- username
- password
- email
- phoneNumber
- firstname
- lastName
- active
- id

Обект „Роля“ - RoleEntity

- name
- roleId

Обект „Спортно игрище“ - Field Entity

- fieldId
- fieldname
- location
- type
- state
- price
- contactInformation
- workingHours

Обект „Резервация“ - Reservation Entitity

- reservationId
- madeBy
- fieldname
- reservationDuration

Хранилища – “Repositories”

Хранилище за потребители - UserRepository

- findByEmail()
- findByUsername()
- findById()
- deleteById()
- findByRolesIn()

Хранилище за роли - RoleRepository

- findByName()

Хранилище за спортни игрища - FieldRepository

- deleteById()
- FindByFieldId()
- FindByFieldName()

Хранилище за резервации - ReservationRepository

- findByFieldNameAndReservationDuration()
- findById()
- findAllByMadeBy()

Услуги (Services)
Услуги за потребители – UserService

- Методи addRegisteredCustomer(), addCompany(), addAdmin()

```
public String addRegisteredCustomer(Users user){
    if(userRepository.findByEmail(user.getEmail()) != null) {
        return "Вече съществува потребител с този имейл!";
    }
    if(userRepository.findByUsername(user.getUsername()) != null) {
        return "Потребителското име е заето!";
    }else {
    Role roleCustomer = roleRepository.findByName("Customer");
    user.addRole(roleCustomer);

    user.setPassword(passwordEncoder.encode(user.getPassword()));
    user.setActive(true);
    userRepository.save(user);
    return "Потребителят е добавен успешно!";
    }
}
```

- Методи viewAllCompanies(), getAllAdmins(), viewAllCustomers()

```
public String viewAllCompanies() {
    if (userRepository.findAll().isEmpty()) {
        return "Няма регистрирани потребители!";
    } else {
        Role roleCompany = roleRepository.findByName("Company");
        Set<Role> searchRoles = new HashSet<>();
        searchRoles.add(roleCompany);
        Iterable<Users> companies =  userRepository.findByRolesIn(searchRoles);
        String companiesToString = "";
        for (Users users : companies) {
            companiesToString = companiesToString + users.toString() + " ";
        }
        if(companiesToString == "") {
            companiesToString = "Няма регистрирани компании!";
        }
        return companiesToString;
    }
}
```

- Метод removeUser()

```
@Transactional
public String removeUser ( long id){
    Users checkIfExists = userRepository.findById(id);
    if (checkIfExists != null) {
        checkIfExists.getRoles().clear();
        userRepository.deleteById(id);
        return "Потребителят е успешно премахнат!";
    } else {
        return "Не съществува потребител с такова id!";
    }
}
```

- Метод changePersonalInformation()

```
public String changePersonalInformation(Users newUserData, String email){

    Users customerToEdit = userRepository.findByEmail(email);
    customerToEdit.setFirstName(newUserData.getFirstName());
    customerToEdit.setLastName(newUserData.getLastName());
    customerToEdit.setEmail(newUserData.getEmail());
    customerToEdit.setPhoneNumber(newUserData.getPhoneNumber());
    customerToEdit.setPassword(passwordEncoder.encode(newUserData.getPassword()));

    userRepository.save(customerToEdit);
    return "Профилната информация беше успешно обновена!";
}
```

- Метод viewProfileInfo()

```
public String viewProfileInfo(String email){
    Users currUser = userRepository.findByEmail(email);
    if(!(currUser == null)) {
        String userData = currUser.toString();
        String reservationsByUser = reservationService.getReservationHistory("{" + currUser.getUsername() + "}");

        return userData + ", " +  reservationsByUser;
    }
    else {
        return "Няма намерен потребител!";
    }

}
```

Услуга за Роли - RoleService

- Методи addRole(), getAllRoles()

```
public void addRole(Role role) {
    if((roleRepository.findAll().contains(role))) {
        throw new IllegalArgumentException("Тази роля вече съществува!");
    }else {
        roleRepository.save(role);
    }
}

public Iterable<Role> getAllRoles(){
    if(roleRepository.findAll() == null){
        System.out.println("Няма добавени роли.");
        return null;
    }
    return roleRepository.findAll();
}
```

Услуга за спортни игрища - FieldService

```
public List<String> getAllFieldsByCity(String city){
    List<String> fieldsForCity = new ArrayList<>();

    if(fieldRepository.findAll() == null){
        fieldsForCity.add("Няма добавени игрища.");
    }else {
        List<Field> allFieldsIterable = fieldRepository.findAll();
        System.out.print(city);
        for (Field field : allFieldsIterable) {
            String adress = field.getLocation();
            if(adress.contains(city)) {
                fieldsForCity.add(field.toString());
                fieldsForCity.add(System.lineSeparator());
            }
        }
        if(fieldsForCity.isEmpty()) {
            fieldsForCity.add("Няма добавени игрища за този град!");
        }
    }
    return fieldsForCity;
}
```

- Метод addNewField()
- Метод deleteField()
- Метод getAllFields()
- Метод getAllFieldsByCity()

- Методи getFieldById(), changeFieldState()

```
public void changeFieldState(int fieldId, String state){
    Field toEdit = fieldRepository.findByFieldId(fieldId);
    String changeStateTo = "";
    if(toEdit.getState().contains(state)) {
        changeStateTo = toEdit.getState().replaceAll(state, "");
        if(changeStateTo.equals("") || changeStateTo.equals(" ")) {
            changeStateTo = "Свободно";
        }
    }
    else if(toEdit.getState().equals("Свободно")) {
        changeStateTo = state;
    }else {
        changeStateTo = toEdit.getState() + " " + state;
    }
    toEdit.setState(changeStateTo);
    fieldRepository.save(toEdit);
}
```

- Метод reserve()
- Метод changeFieldInfo()
- Метод getAllFieldsByType()

```
public String reserve(String madeBy, int fieldId, String duration) {
    Field fieldToReserve = fieldRepository.findByFieldId(fieldId);
    Reservation reservation = new Reservation();

    if(fieldToReserve.getState().contains(duration)){
        return "Игрището вече е резервирано за този период. Моля изберете друг.";
    }else {
        reservation.setFieldName(fieldToReserve.getFieldName());
        reservation.setMadeBy(madeBy);
        reservation.setReservationDuration(duration);
        reservationRepository.save(reservation);
        long reservationId = reservationRepository
                .findByFieldNameAndReservationDuration(fieldToReserve.getFieldName(), duration).getId();

        String newState = String.format("Резервирано за " + duration);
        changeFieldState(fieldId, newState);

        return String.format("Игрището %s е резервирано от %s за периода %s. Вашият номер на резервацията е %d.",
                fieldToReserve.getFieldName(), madeBy, duration,reservationId);
    }

}
```

Услуга за резервации - ReservationService

```
public String reserveField(String madeBy, int id, String duration) {
    return fieldService.reserve(madeBy, id, duration);
}
```

```
public String cancelReservation(long reservationId, int fieldId) {
        Reservation toCancel = reservationRepository.findById(reservationId);
        if(toCancel == null) {
            return "Няма резервация с този номер!";
        }else {
            String reservationPeriod = toCancel.getReservationDuration();
            String stateToRemove = "Резервирано за " + reservationPeriod;
            fieldService.changeFieldState(fieldId, stateToRemove);
            reservationRepository.delete(toCancel);
            return "Резервацията е успешно отменена!";
        }
    }
```

- Методи:
  - reserveField()
  - cancelReservation()
  - getReservationHistory()
  - viewAllReservations()

Управление (Controllers)
Основно управление - MainController

```java
@GetMapping("/view_all_fields")
public List<String> getAllFields(){
    return fieldService.getAllFields();
}

@GetMapping("/view_all_fields_for_city/{city}")
public List<String> getAllFieldsForCity(@PathVariable String city){
    return fieldService.getAllFieldsByCity(city);
}

@GetMapping("/view_all_fields_for_type/{type}")
public List<String> getAllFieldsForType(@PathVariable String type){
    return fieldService.getAllFieldsByType(type);
}

@PostMapping("/add_role")
public String addNewRole(@RequestBody Role role) {
    roleService.addRole(role);
    return "Ролята е успешно добавена!!";
}

@GetMapping("/get_all_roles")
public Iterable<Role> getAllRoles(){
    return roleService.getAllRoles();
}
```

Потребителското управление – UserController част 1

```java
@RequestMapping(value = "/view_profile_info", method = RequestMethod.GET)
@ResponseBody
public String customerProfile(Principal principal){
    String email = principal.getName();
    System.out.print(email);
    return userService.viewProfileInfo(email);
}

@PutMapping("/change_user_information/{email}")
public String changeUserInformation(@RequestBody Users newCompanyData, @PathVariable String email){
    return userService.changePersonalInformation(newCompanyData, email);
}

@DeleteMapping("/delete_field/{fieldId}")
public String removeField(@PathVariable int fieldId){
    fieldService.deleteField(fieldId);
    return String.format("Игрище с идентификационен номер %d е изтрито!", fieldId);
}

@PutMapping("/reserve_field/{madeBy}/{fieldId}")
public String reserveField(@PathVariable String madeBy, @PathVariable int fieldId, @RequestBody String duration){
    return reservationService.reserveField(madeBy, fieldId, duration);
}

@DeleteMapping("/cancel_reservation/{reservationId}/{fieldId}")
public String cancelReservation(@PathVariable long reservationId,@PathVariable int fieldId){
    return reservationService.cancelReservation(reservationId, fieldId);
}
```

Потребителското управление – UserController част 2

```java
@GetMapping("/admin/view_all_companies")
public String viewAllCompanies() {
        return userService.viewAllCompanies();
}

@GetMapping("/admin/view_all_reservations")
public String viewAllReservations(){
    return reservationService.viewAllReservations();
}

@GetMapping("/admin/view_all_customers")
public String viewAllCustomers(){
    return userService.viewAllCustomers();
}

@GetMapping("/admin/get_all_admins")
public String viewAllAdmins(){
    return userService.getAllAdmins();
}

@Transactional
@DeleteMapping("/admin/delete_user/{userId}")
public String deleteUser(@PathVariable long userId){
        return userService.removeUser(userId);
}

@PostMapping("/admin/register_admin")
public String addNewAdmin(@RequestBody Users user){
    String resultString = userService.addAdmin(user);
    return resultString;
}
```

Потребителското управление – UserController част 3


```
@GetMapping("/admin/home")
public String adminHomePage(){
    String welcome = "Добре дошли във вашият администраторски профил профил!\n\n";
    String menuString = "Меню: \n\n";
    String profile =
            "<HTML><body> <a href=\"http://localhost:8080/view_profile_info\">Преглед на профилни данни</a></body></HTML>";
    String editProfile =
            "<HTML><body> <a href=\"http://localhost:8080/change_user_information/{email}\">Промяна на профилни данни</a></body></HTML>";
    String removeUser =
            "<HTML><body> <a href=\"http://localhost:8080/admin/delete_user/{companyId}\">Изтриване на потребител</a></body></HTML>";
    String removeFields =
            "<HTML><body> <a href=\"http://localhost:8080/delete_field/{fieldId}\">Изтриване на игрище</a></body></HTML>";
    String cancelReservation =
            "<HTML><body> <a href=\"http://localhost:8080/cancel_reservation/{reservationId}/{fieldId}\">Отмяна на резервация</a></body></HTML>";
    String viewAllReservation =
            "<HTML><body> <a href=\"http://localhost:8080/admin/view_all_reservations\">Преглед на всички резервации</a></body></HTML>";
    String viewAllCustomers =
            "<HTML><body> <a href=\"http://localhost:8080/admin/view_all_customers\">Преглед на всички клиенти</a></body></HTML>";
    String viewAllCompanies =
            "<HTML><body> <a href=\"http://localhost:8080/admin/view_all_companies\">Преглед на всички компании</a></body></HTML>";
    String viewAllAdmins =
            "<HTML><body> <a href=\"http://localhost:8080/admin/get_all_admins\">Преглед на всички админи</a></body></HTML>";
    String addNewAdmins =
            "<HTML><body> <a href=\"http://localhost:8080/admin/register_admin\">Преглед на всички админи</a></body></HTML>";
```

```java
@PutMapping("/company/change_field_information/{id}")
public String changeFieldInformation(@RequestBody Field newData, @PathVariable int id){
    return fieldService.changeFieldInfo(newData, id);
}

@PostMapping("/company/add_new_field")
public String addNewField(@RequestBody Field field) {
    return fieldService.addNewField(field);
}
```

```java
@PostMapping("/company_add_company")
public String addCompany(@RequestBody Users user){
    String statusString = userService.addCompany(user);
    return statusString;
}

@PostMapping("/customer_register_customer")
public String addNewCustomer(@RequestBody Users user){
    String resultString =  userService.addRegisteredCustomer(user);
    return resultString;
}
```

Защита на приложението и нейната конфигурация


```java
public class CustomUserDetails implements UserDetails {

    private Users user;

    public CustomUserDetails(Users user) {
        this.user = user;
    }

    @Override
    public Collection<? extends GrantedAuthority> getAuthorities() {
        Set<Role> roles = user.getRoles();
        System.out.println(roles);
        List<SimpleGrantedAuthority> authorities = new ArrayList<>();

        for (Role role : roles) {
            authorities.add(new SimpleGrantedAuthority(role.getName()));
        }

        return authorities;
    }
```

```java
    @Override
    public String getPassword() {
        return user.getPassword();
    }

    @Override
    public String getUsername() {
        return user.getEmail();
    }

    @Override
    public boolean isAccountNonExpired() {
        return true;
    }

    @Override
    public boolean isAccountNonLocked() {
        return true;
    }

    @Override
    public boolean isCredentialsNonExpired() {
        return true;
    }

    @Override
    public boolean isEnabled() {
        return user.isActive();
    }
```

Клас CustomUserDetails()

Клас CustomUserDetailsService()


```
@Service
@Transactional
public class CustomUserDetailsService implements UserDetailsService {

    @Autowired
    private UserRepository userRepository;

    @Override
    public UserDetails loadUserByUsername(String email)
            throws UsernameNotFoundException {
        Users user = userRepository.findByEmail(email);

        if (user == null) {
            throw new UsernameNotFoundException("Could not find user");
        }

        return new CustomUserDetails(user);
    }

```

Клас Конфигурация на сигурността - SecurityConfig


```java
@Override
public void configure(WebSecurity web) throws Exception {
    web.ignoring()
    .antMatchers("/home")
    .antMatchers("/view_all_fields")
    .antMatchers("/view_all_fields_for_city/{city}")
    .antMatchers("/view_all_fields_for_type/{type}")
    .antMatchers("/company_add_company")
    .antMatchers("/customer_register_customer");
}

@Override
protected void configure(AuthenticationManagerBuilder auth) throws Exception
    auth.authenticationProvider(authenticationProvider());
}
```

```java
@Configuration
@EnableWebSecurity
public class SecurityConfig extends WebSecurityConfigurerAdapter {

    @Bean
    public UserDetailsService usersDetailsService() {
        return new CustomUserDetailsService();
    }
    @Bean
    public PasswordEncoder passwordEncoder() {
        return new BCryptPasswordEncoder();
    }
    @Bean

    public DaoAuthenticationProvider authenticationProvider() {
        DaoAuthenticationProvider authenticationProvider = new DaoAuthenticationProvider();
        authenticationProvider.setUserDetailsService(usersDetailsService());
        authenticationProvider.setPasswordEncoder(passwordEncoder());
        return authenticationProvider;
    }
```

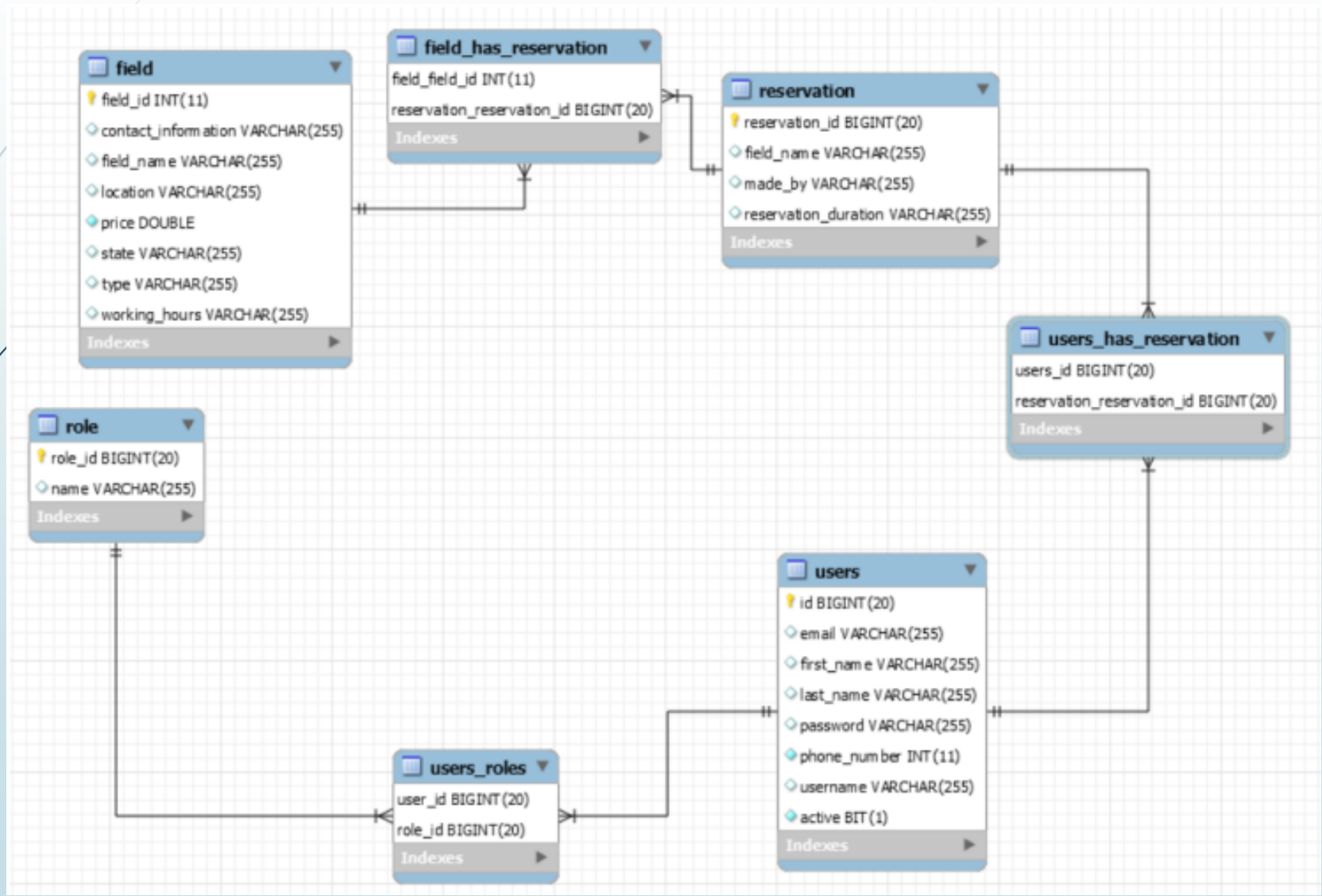
```java
@Override
protected void configure(HttpSecurity http) throws Exception {

    http.authorizeRequests()
    .antMatchers("/home").permitAll()
    .antMatchers("/view_profile_info", "/change_user_information/{email}").hasAnyAuthority("Admin", "Customer",
    .antMatchers("/reserve_field/{madeBy}/{fieldId}", "/cancel_reservation/{reservationId}/{fieldId}")
    .hasAnyAuthority("Admin", "Customer")
    .antMatchers("/delete_field/{fieldId}").hasAnyAuthority("Admin", "Company")
    .antMatchers("/admin/**", "add_role", "get_all_roles").hasAuthority("Admin")
    .antMatchers("/customer/**").hasAuthority("Customer")
    .antMatchers("/company/**").hasAuthority("Company")
    .anyRequest().authenticated()
    .and()
    .formLogin().permitAll()
    .and()
    .logout().permitAll();
}
```

Файл с настройки „ApplicationProperties“ на приложението

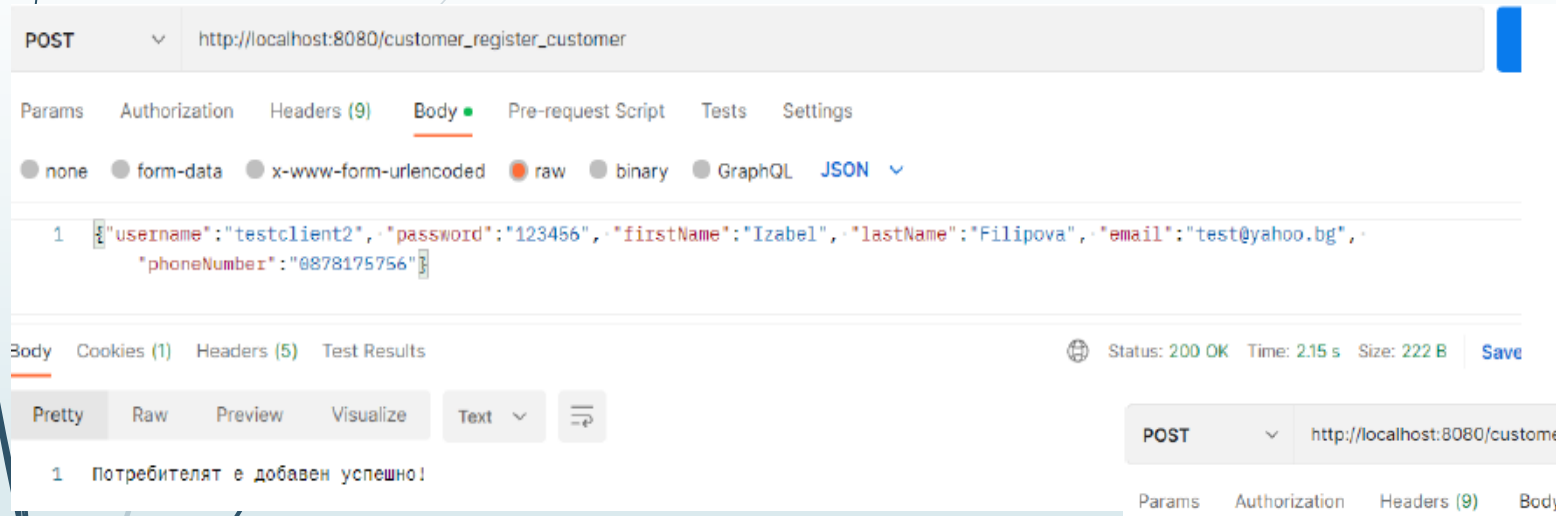
```
spring.datasource.url=jdbc:mysql://127.0.0.1:3306/playingfieldreservations?useSSL=false
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL5InnoDBDialect
spring.datasource.username=root
spring.datasource.password=51!wkv-4!t
spring.jpa.hibernate.ddl-auto=update
spring.jpa.properties.hibernate.globally_quoted_identifiers=true
```

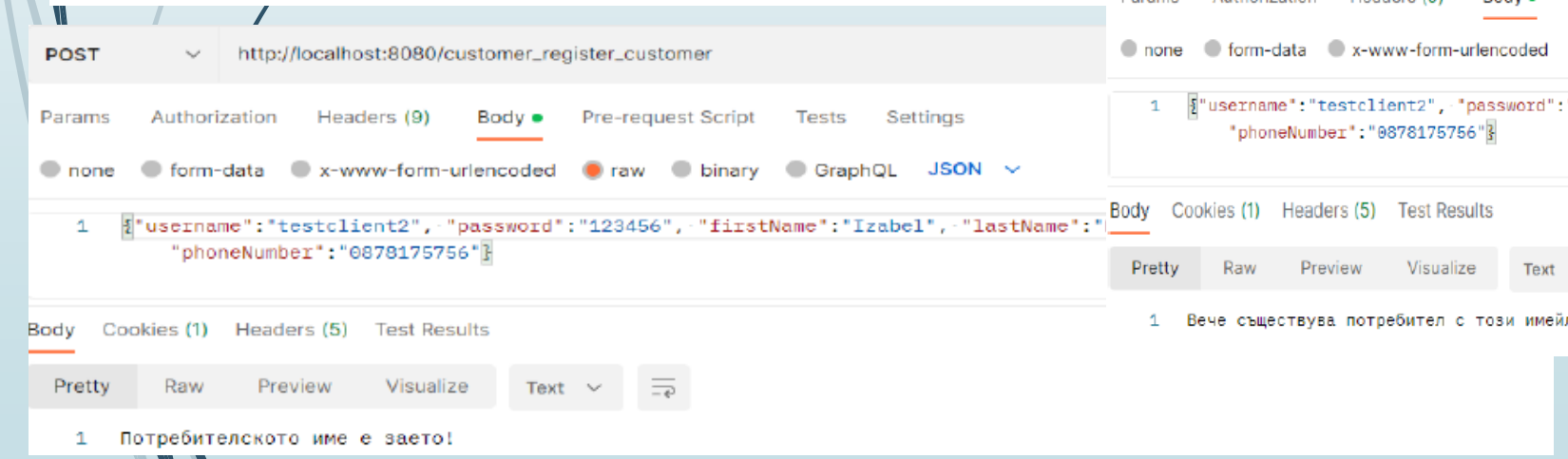
Диаграма на връзките между обектите (ER – Entity Relationship diagram)

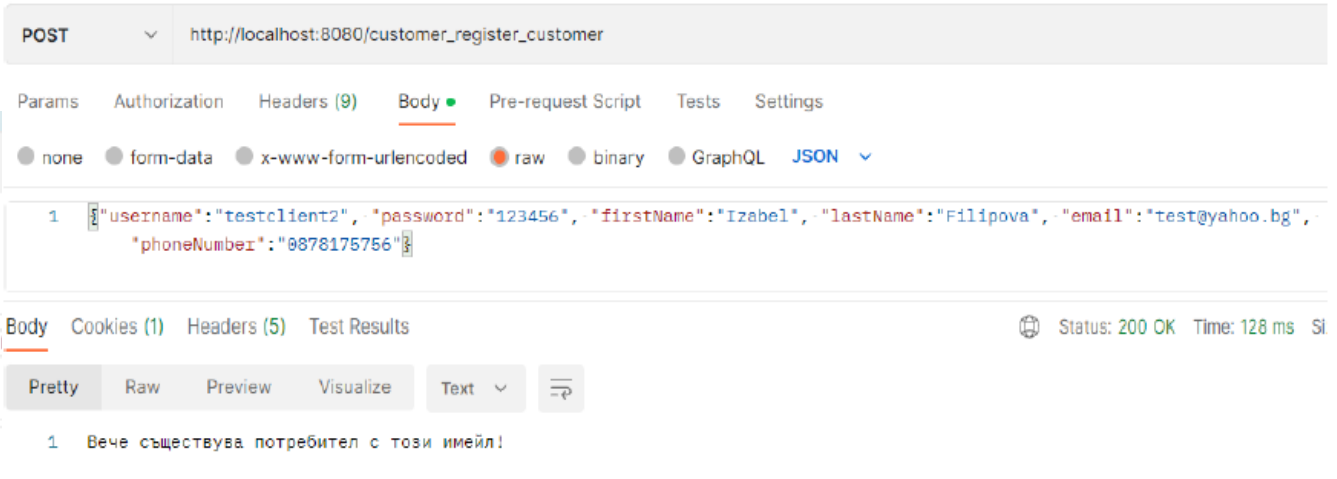


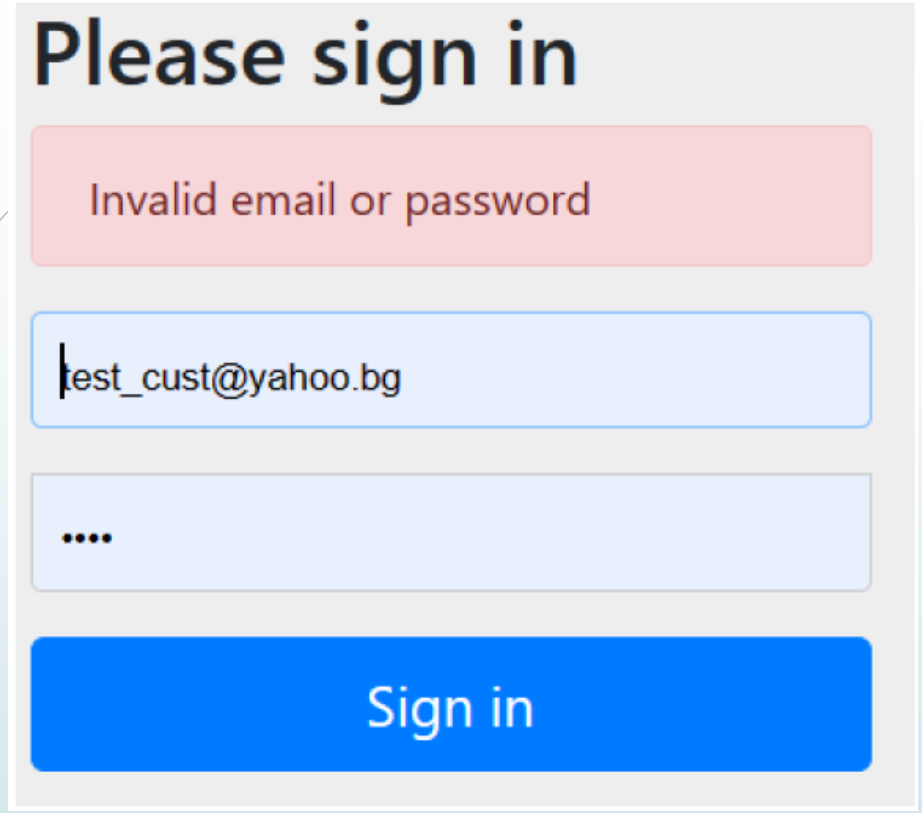
Експериментални данни и изводи

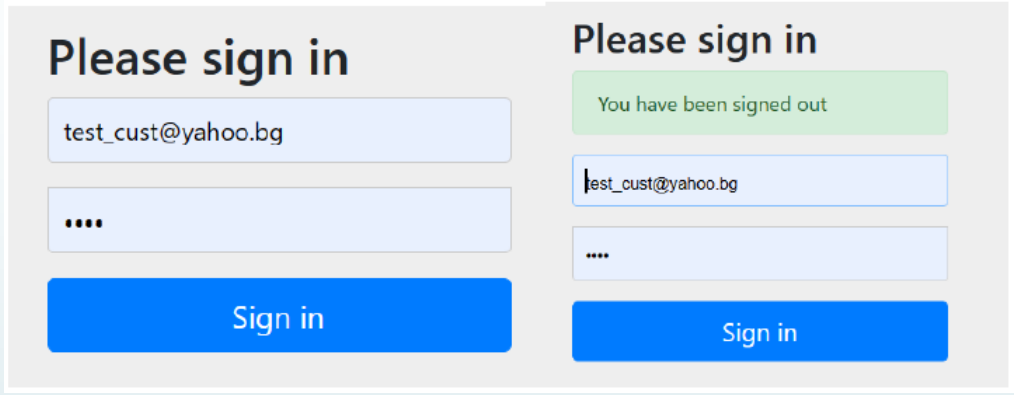
- Експериментални данни
  - Добавяне на нов потребител





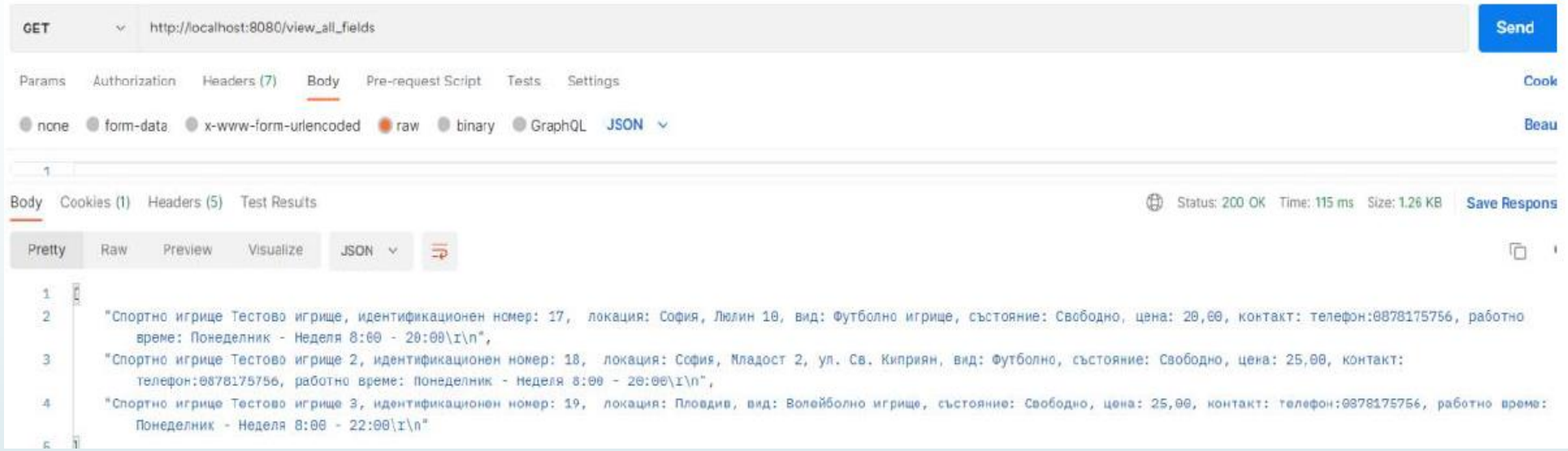


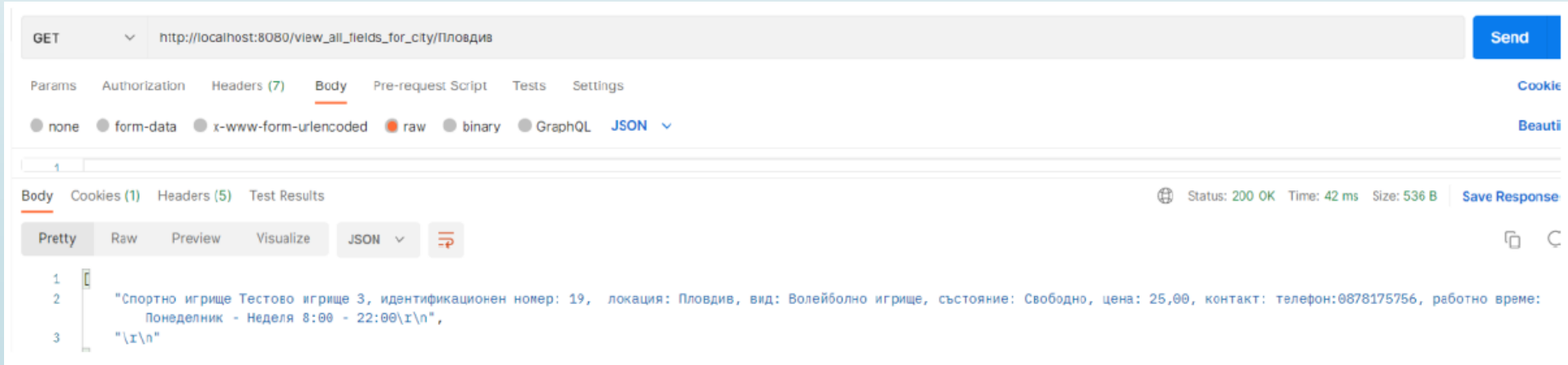




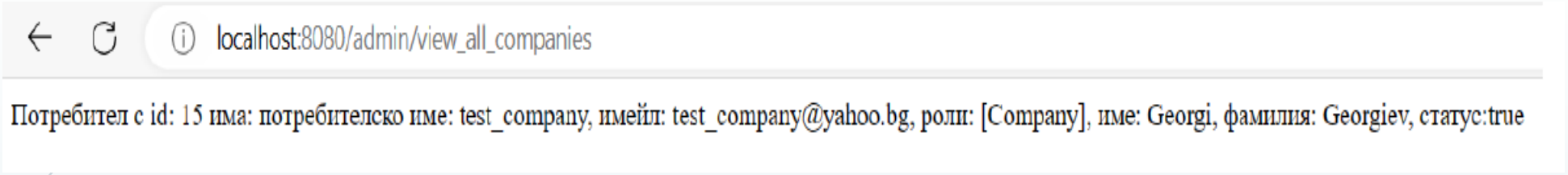
Вход/изход в приложението чрез лог-ин форма „formLogin()“

Преглед на спортни игрища

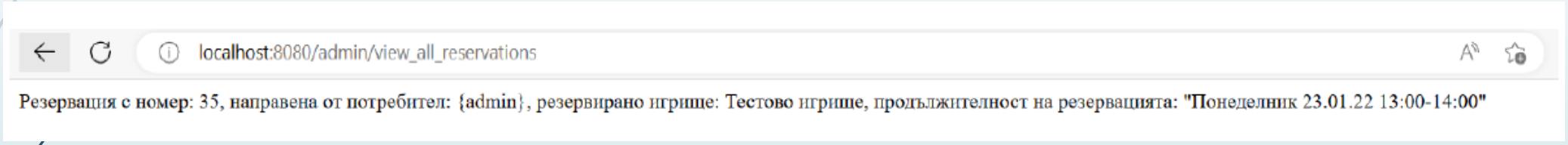




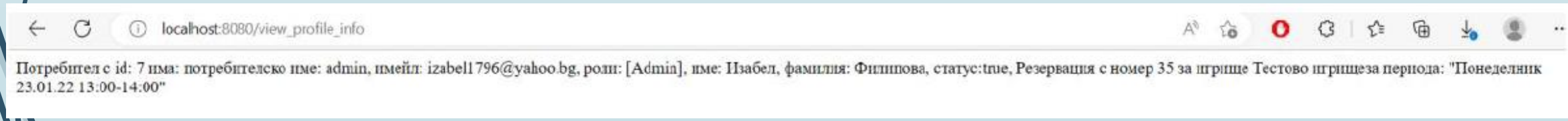
- Достъп на админ до потребителите от тип компания



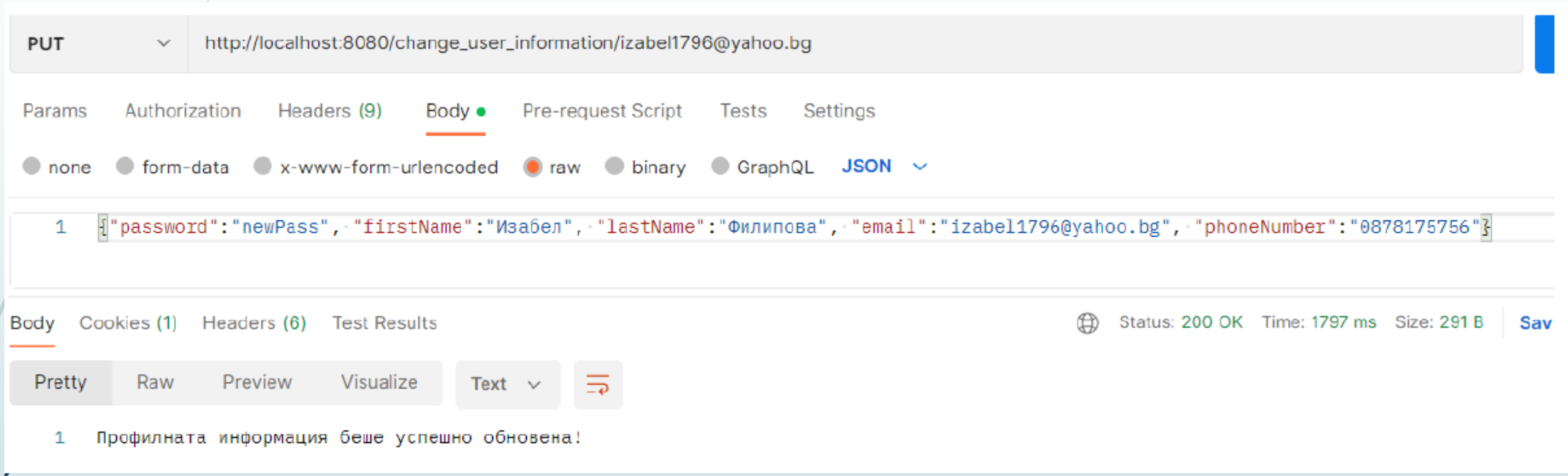
- Преглед на всички резервации от админ



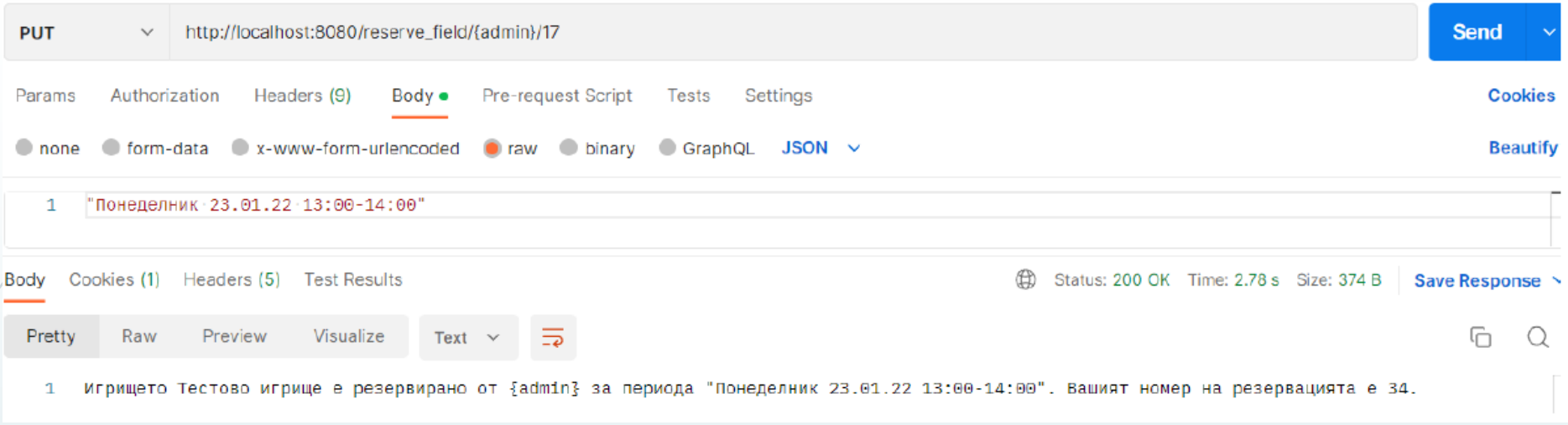
- Преглед на профилни данни и направени резервации

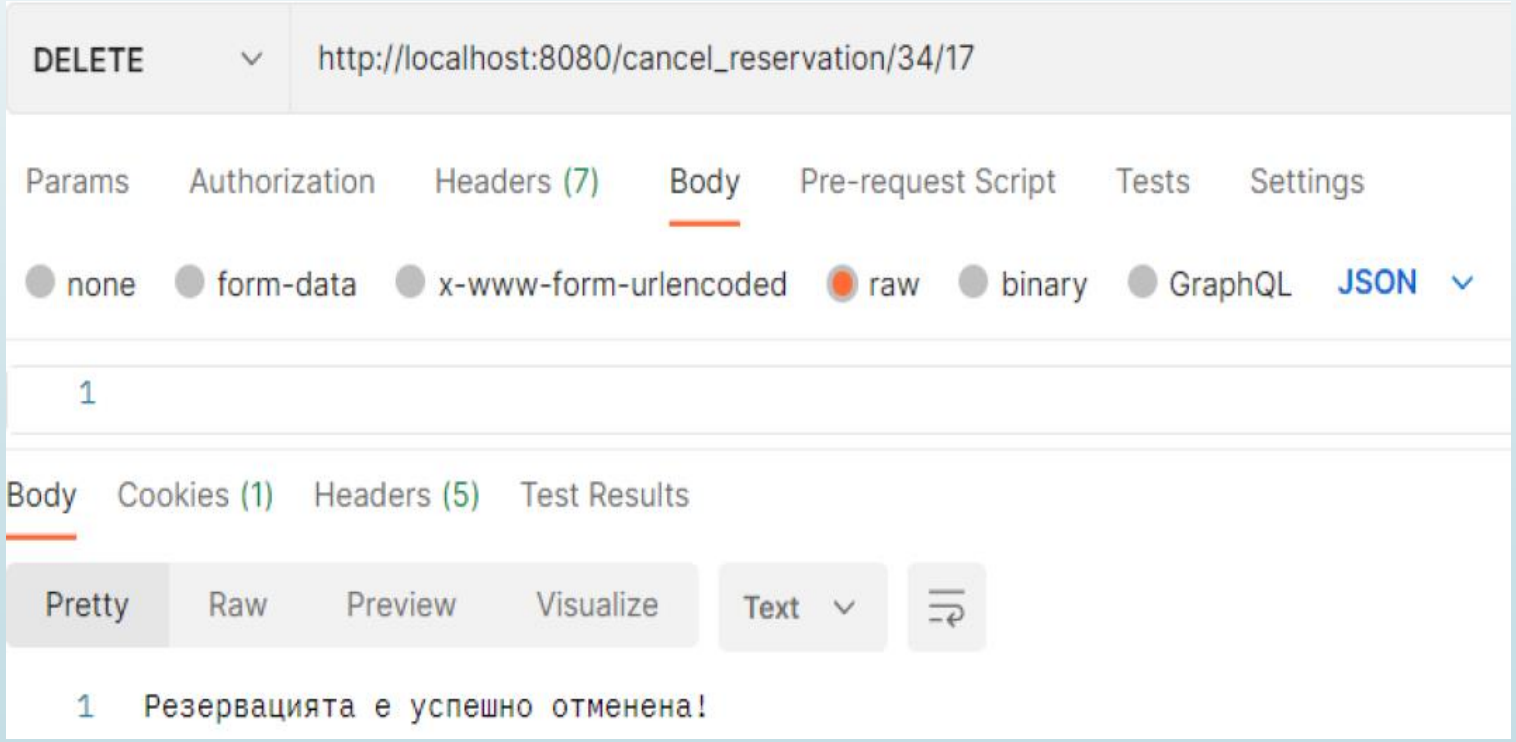


Ъпдейт на профилни данни

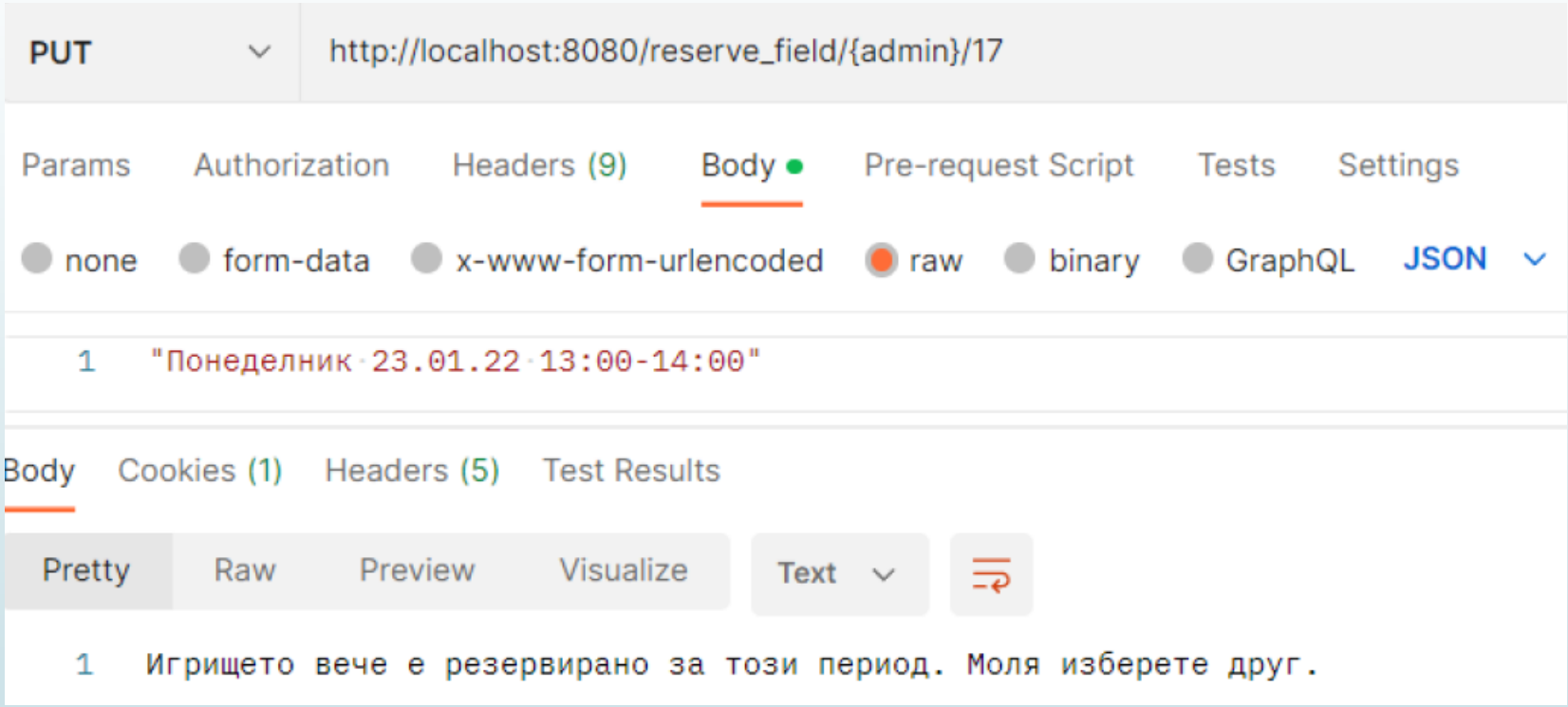


Резервация/отмяна на резервация

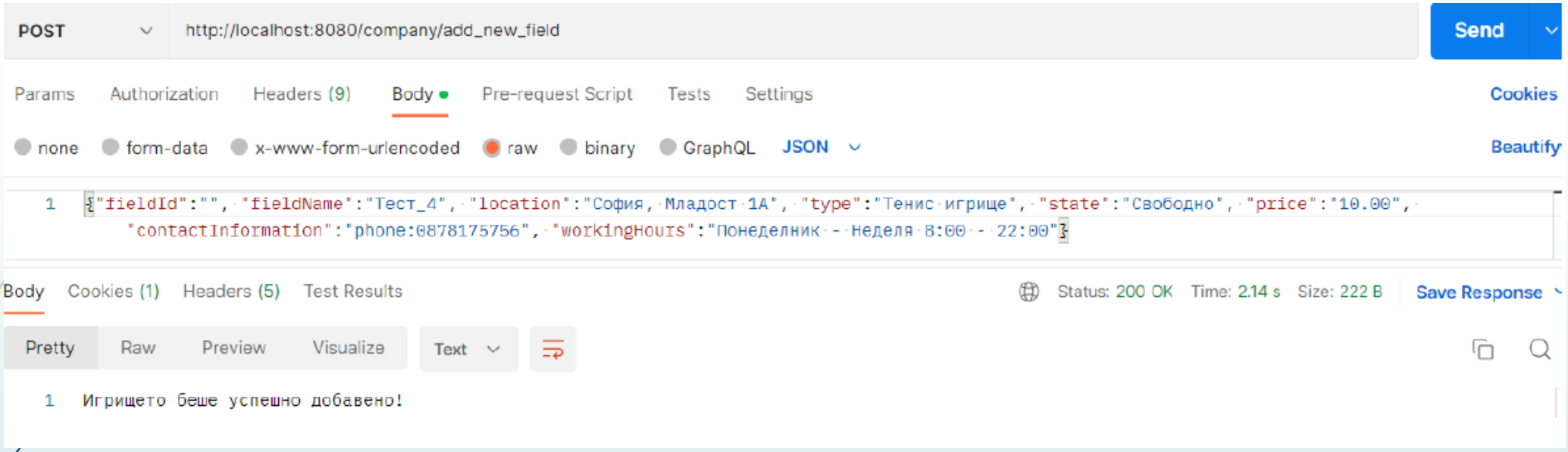


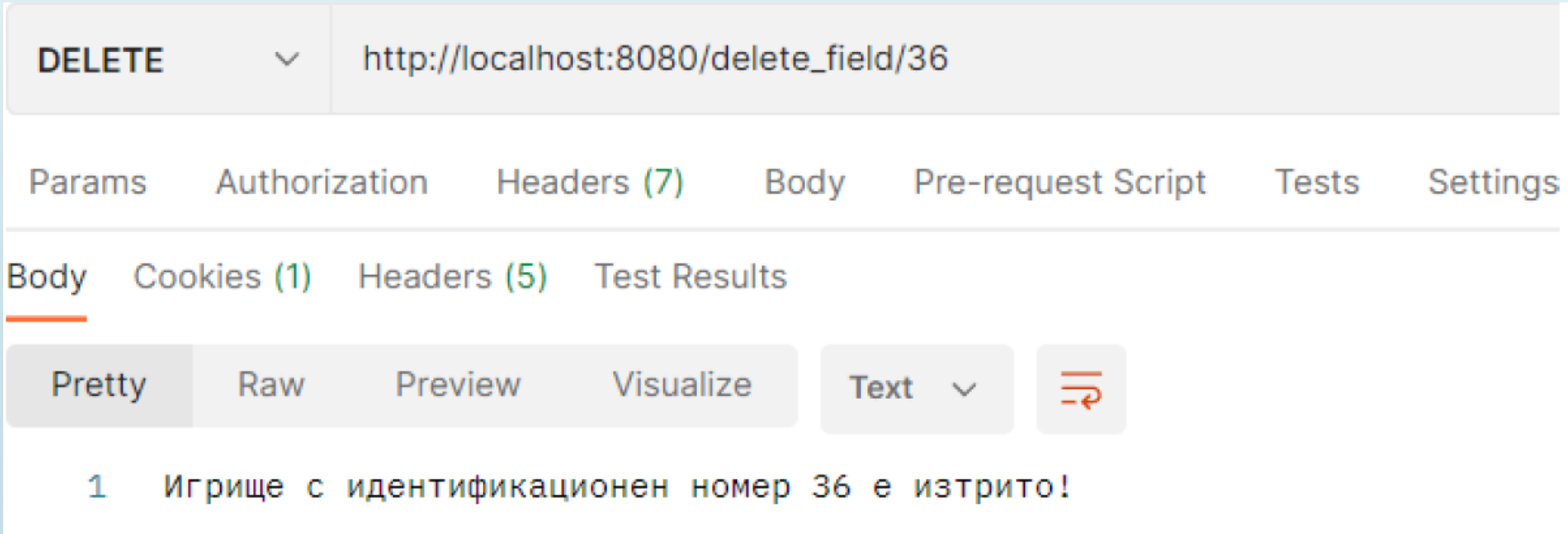


Зает период за резервация



Добавяне/премахване на игрище





Изводи и бъдеща оптимизация

- Зададените технологии са използвани за програмната реализация
- Подобренията на дипломният проект са спазени
  - Свободна резервация/отмяна
  - Обхват на игрища в цялата страна
  - Свободен преглед на спортни игрища
  - Достъп до история на резервациите
- Варианти за оптимизация
  - Фронт-енд интерфейс с модул за график
  - Интеграция на „Google maps“
  - Чат платформа

Благодаря за вниманието!