МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение
высшего образования «Самарский национальный исследовательский университет
имени академика С.П. Королева»
(Самарский университет)

Институт информатики и кибернетики
Кафедра технической кибернетики

ОТЧЕТ

по лабораторной работе №1
Дисциплина «Технологии сетевого программирования»

Выполнили студенты
группы 6301-010302D
Жданова В.И.
Русяева С.П.

САМАРА 2025

Создан Docker-контейнер для PostgreSQL
В application.properties настроены параметры подключения к PostgreSQL

Созданы сущности для основных моделей User, Cart, CartItem, Order, OrderItem, Product, Brand
Пример сущности User:

```
@Data
@Entity
@Table(name = "users")
@FieldDefaults(level = AccessLevel.PRIVATE)
public class Users {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    Long userId;

    @NotBlank(message = "Username can't be blank")
    String userName;

    @NotBlank(message = "Email can't be blank")
    @Column(unique = true)
    String email;

    @NotBlank(message = "Password can't be blank")
    String password;

    String phone;

    @OneToMany(mappedBy = "user", cascade = CascadeType.ALL)
    @JsonIgnore
    List<Orders> orders;

    @OneToOne(mappedBy = "user", cascade = CascadeType.ALL)
    @JsonIgnore
    Cart cart;
}
```

Созданы репозитории для взаимодействия с бд

Созданы DTO для передачи данных между клиентом и сервером
Пример DTO для User:

```
@Data
@FieldDefaults(level = AccessLevel.PRIVATE)
public class UsersDto {
   String userName;
   String email;
```

```
  String password;
  String phone;
}
```

Для каждой сущности созданы сервисы, реализующие операции создания, чтения, удаления

Пример сервиса для User:

```
@Service
@RequiredArgsConstructor
@FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
public class UsersService {

  UsersRepository usersRepository;
  CartRepository cartRepository;
  Mapper mapper;
  PasswordEncoder passwordEncoder;

  public List<Users> findAll() { return usersRepository.findAll(); }

  public Users save(UsersDto userDto) {
     Users user = mapper.toUsers(userDto);
     user.setPassword(passwordEncoder.encode(user.getPassword()));
     return usersRepository.save(user);
  }

  public void delete(Long userId) {
     usersRepository.deleteById(userId);
  }
}
```

Для каждой сущности созданы REST-контроллеры.

Пример контроллера для User:

```
@FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
@RestController
@RequiredArgsConstructor
@RequestMapping("/api/shop/users")
public class UsersController {

  UsersService usersService;

  @GetMapping()
  public ResponseEntity<List<Users>> getUsers() {
     return ResponseEntity.ok(usersService.findAll());
  }

  @PostMapping()
```

```
  public ResponseEntity<Users> addUsers(@Valid @RequestBody UsersDto users) {
    return ResponseEntity.status(HttpStatus.CREATED).body(usersService.save(users));
  }

  @DeleteMapping("/{userId}")
  public ResponseEntity<?> deleteUsers (@PathVariable Long userId) {
    usersService.delete(userId);
    return ResponseEntity.ok(String.format("User %s deleted", userId));
  }
}
```

Для сущности Product также реализованы функции сортировки по цене и названию:

```
public List<Product> getProductsSortedByNameAsc() {
  return productRepository.findAll(Sort.by(Sort.Direction.ASC, "name"));
}

public List<Product> getProductsSortedByNameDesc() {
  return productRepository.findAll(Sort.by(Sort.Direction.DESC, "name"));
}

public List<Product> getProductsSortedByPriceAsc() {
  return productRepository.findAll(Sort.by(Sort.Direction.ASC, "price"));
}

public List<Product> getProductsSortedByPriceDesc() {
  return productRepository.findAll(Sort.by(Sort.Direction.DESC, "price"));
}
```

Использован BCryptPasswordEncoder для хеширования паролей

```
@Configuration
public class SecurityConfig {

  @Bean
  public PasswordEncoder passwordEncoder() {
    return new BCryptPasswordEncoder();
  }
}
```

Использован Flyway для управления миграциями базы данных.
Созданы файлы миграций в src/main/resources/db/migration.
Пример миграции V2__insert-values.sql:

```
INSERT INTO users (user_name, email, password, phone) VALUES ('Ann', 'ann@gmail.com', '123',
'1234567890');
```

```
INSERT INTO users (user_name, email, password, phone) VALUES ('Milana', 'milana@gmail.com',
'456', '0987654321');

INSERT INTO brand (name, country) VALUES ('Clinique', 'France');
INSERT INTO brand (name, country) VALUES ('Loreal', 'France');

INSERT INTO product (name, description, category, brand_id, price) VALUES ('lipstick', 'matt',
'Category1', 1, 1999.99);
INSERT INTO product (name, description, category, brand_id, price) VALUES ('cream', 'hand cream',
'Category2', 2, 299.99);
```

Протестированы эндпоинты с помощью Postman
Пример запроса на добавление бренда:

POST /api/shop/brand

```
{
    "name": "Clinique",
    "country": "France"
}
```

Пример ответа:

```
{
    "brandId": 2,
    "name": "Clinique",
    "country": "France"
}
```

Пример запроса для сортировки продуктов:
GET /api/shop/products/sortByPrice?order=asc