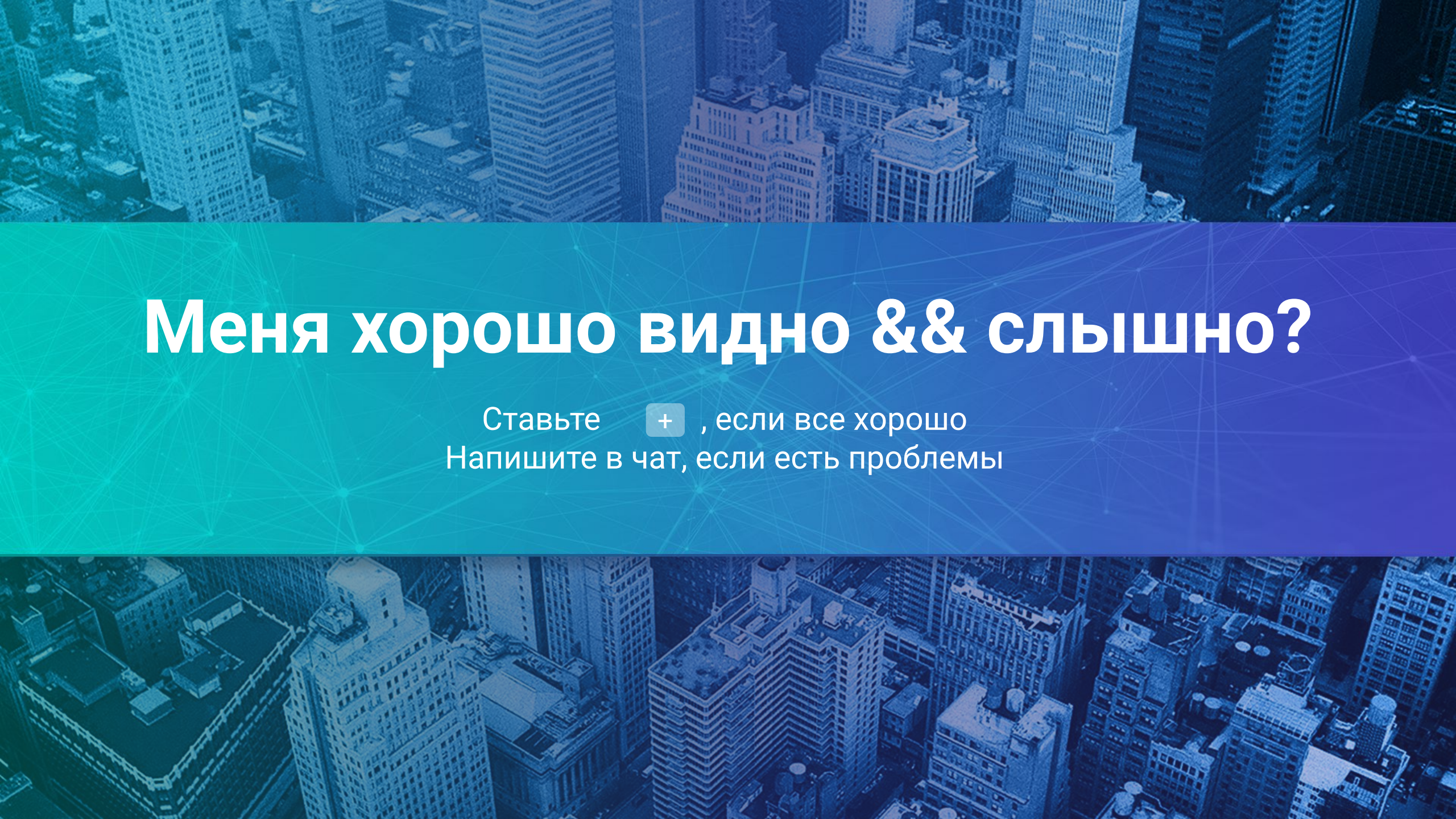

OTUS
ОНЛАЙН-ОБРАЗОВАНИЕ
Онлайн-образование



Меня хорошо видно && слышно?

Ставьте + , если все хорошо
Напишите в чат, если есть проблемы

Защита проекта
Тема: Система онлайн-бронирования репетиций

Белов Василий

Backend-разработчик

belovi-tech@mail.ru

План защиты

- Цели проекта
- Что планировалось
- Используемые технологии
- Что получилось
- Схемы/архитектура
- Выводы

Цели проекта

1. Создать базовый рабочий экземпляр приложения на основе Spring (применить полученные на курсе навыки)

2. Построить приложение на микросервисной архитектуре

3. Получить новые знания вне рамок курса

Что планировалось

1. На Spring’е до курса не писал
2. Был небольшой опыт написания микросервисов, читал про их архитектуру
3. Было желание опробовать Spring при написании близкого к реальности проекта
4. Было представление о предметной области и функционале, который надо реализовать
5. Проект писался с нуля после выполнения дз, 2-3 недели

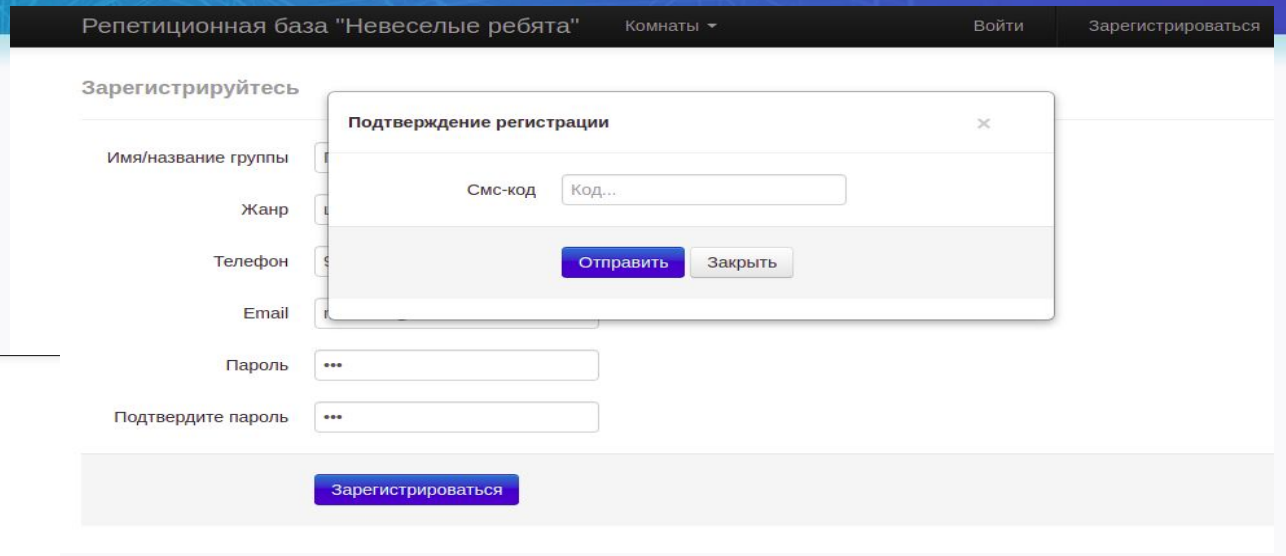
Используемые технологии

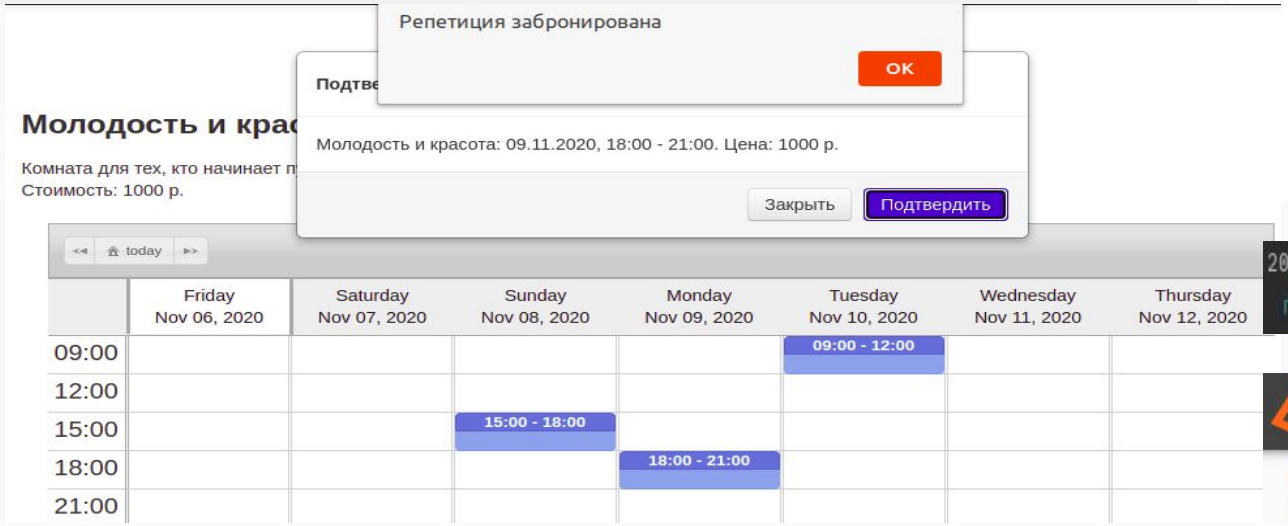
1. Spring Boot 2.3.3, Spring Integration
2. Postgres, Flyway, Spring Data Jpa
3. SpringSecurity OAuth
4. SpringCloud, Netflix
5. Thymeleaf, JQuery :)
   Docker, docker-compose

Что получилось

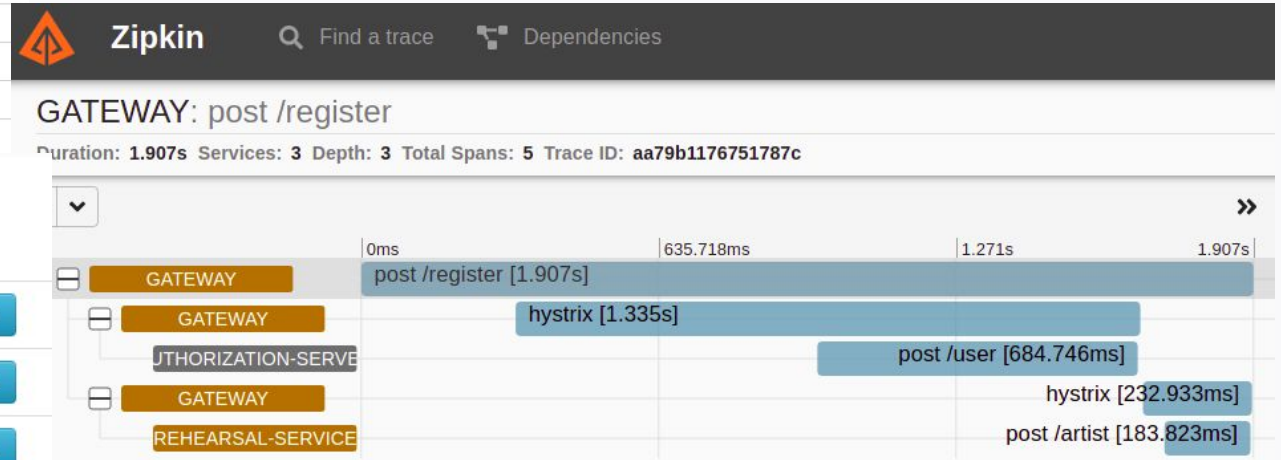
Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
AUTHORIZATION-SERVER	n/a (1)	(1)	UP (1) - 192.168.1.73:authorization-server:8090
GATEWAY	n/a (1)	(1)	UP (1) - 192.168.1.73:gateway:8080
REHEARSAL-SERVICE	n/a (1)	(1)	UP (1) - 192.168.1.73:rehearsal-service:8888
SERVICE-DISCOVERY	n/a (1)	(1)	UP (1) - 192.168.1.73:service-discovery:8761
SMS-SERVICE	n/a (1)	(1)	UP (1) - 192.168.1.73:sms-service:8085





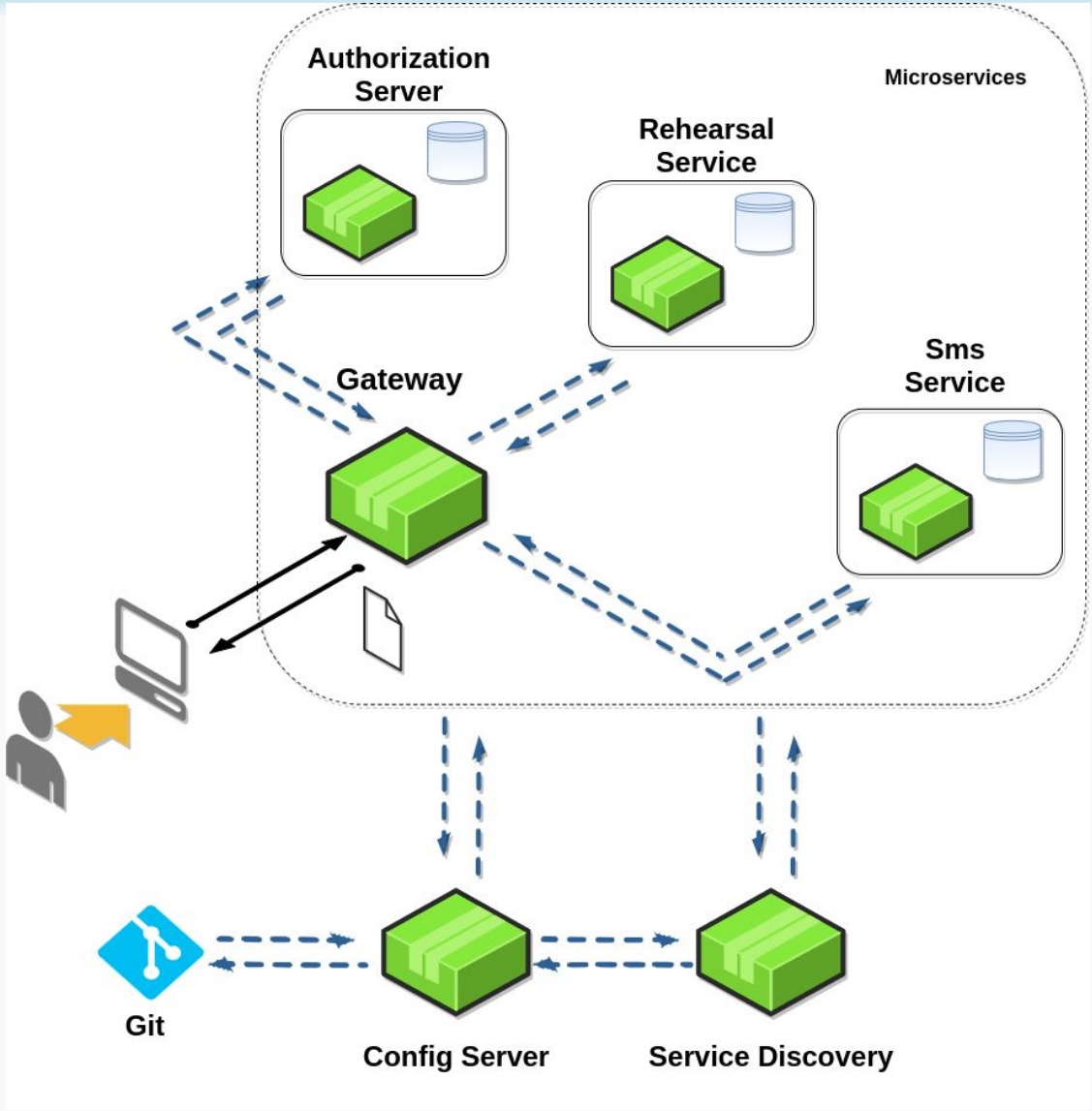

```
2020-11-06 19:03:31.358  INFO [sms-service,3f49dc573be3f1ed,228479f41d67dfd0,true] 33480 --- [     executor-1]
r.o.p.s.s.m.MobileProviderStub          : Sms '6396' was sent to 9011231231
```



Мои репетиции

Дата	Время	Комната	Статус	Цена	Действия
10.11.2020	12:00:00 - 15:00:00	Мама-анархия	RESERVED	780	Отменить
09.11.2020	18:00:00 - 21:00:00	Молодость и красота	RESERVED	1000	Отменить
08.11.2020	14:00:00 - 15:00:00	До-ми-соль	RESERVED	200	Отменить

Схемы (архитектура, БД)



Таблицы базы сервиса AuthorizationServer

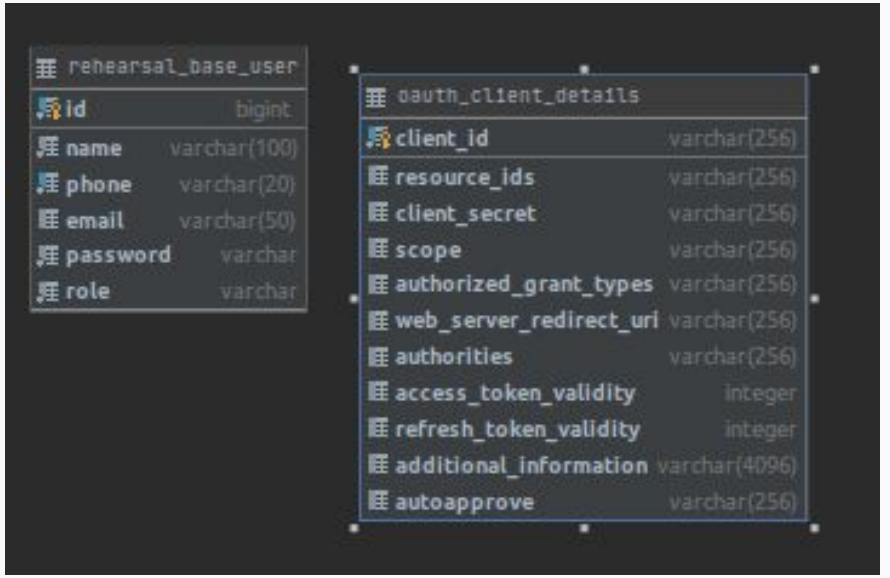
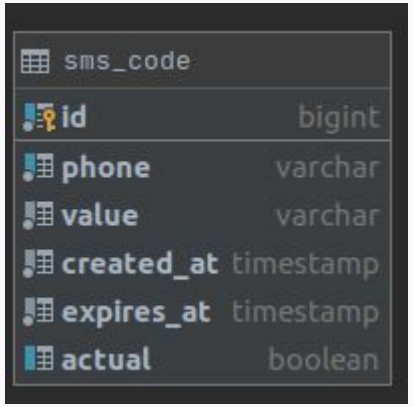
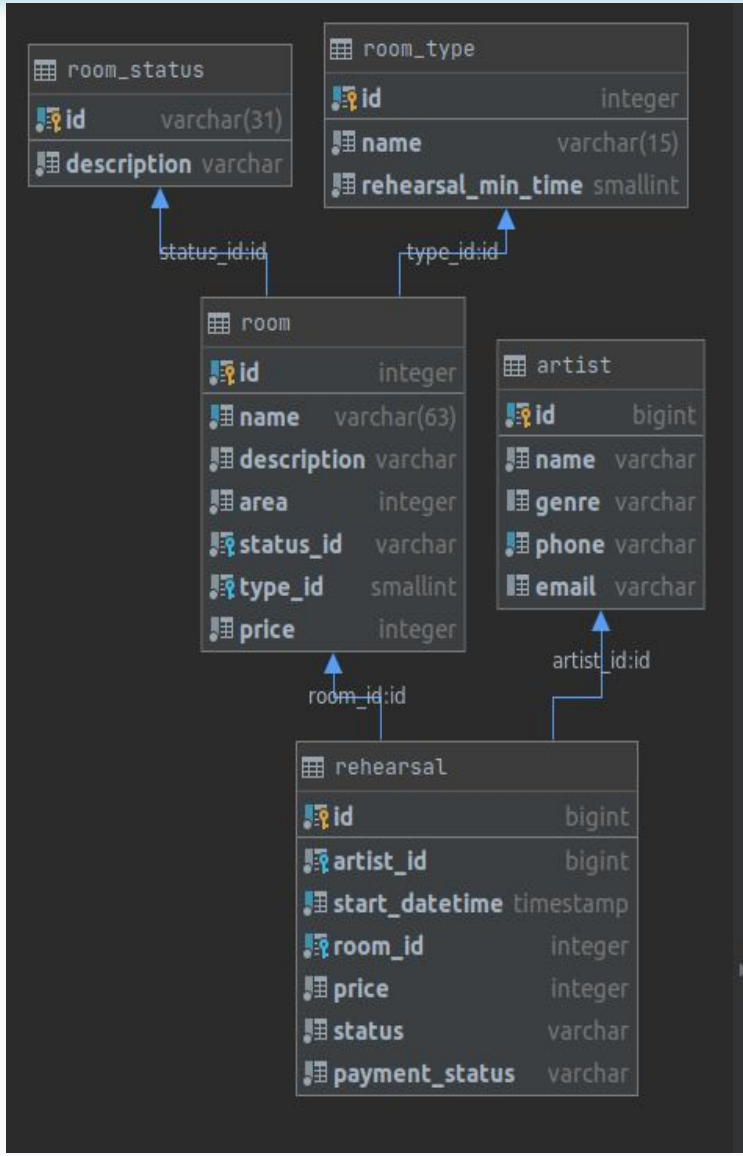


Таблица смс-кодов базы сервиса SmsService



Схемы (архитектура, БД)

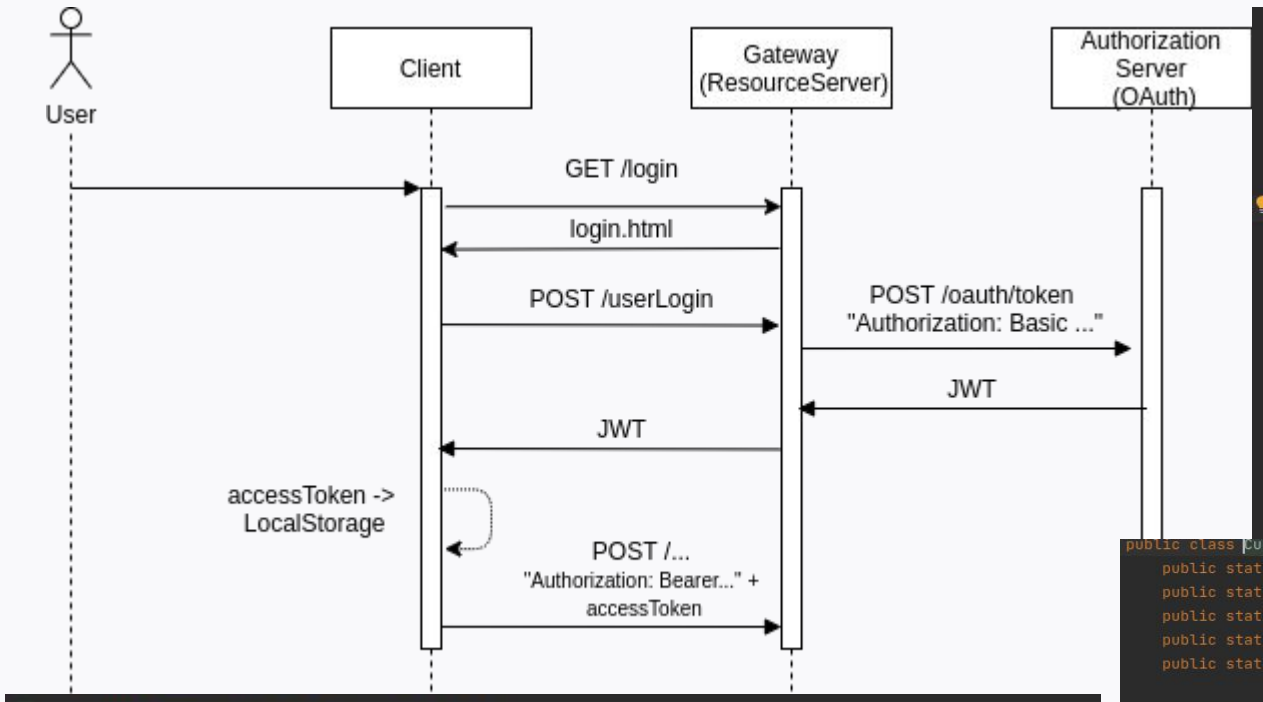



```java
@Entity
@Table(name = "rehearsal")
@NoArgsConstructor
@AllArgsConstructor
@Data
@ToString
@Accessors(chain = true)
public class Rehearsal {
    @Id
    @GeneratedValue(strategy = GenerationTy
    @Column(name = "id")
    private long id;

    @OneToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "artist_id")
    @Setter
    private @NonNull Artist artist;

    @Column(name = "start_datetime")
    private LocalDateTime startDatetime;

    @OneToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "room_id")
    private @NonNull Room room;

    @Column(name = "price")
    private int price;

    @Enumerated(EnumType.STRING)
    private RehearsalStatus status = Rehear

    @Enumerated(EnumType.STRING)
    private PaymentStatus paymentStatus = F
```

```java
@Entity
@Table(name = "room")
@NoArgsConstructor
@AllArgsConstructor
@Data
@ToString
public class Room {
    @Id
    @GeneratedValue(strategy = GenerationT
    @Column(name = "id")
    private int id;

    @Column(name = "name")
    private String name;

    @Column(name = "description")
    private String description;

    @OneToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "status_id")
    private RoomStatus status;

    @OneToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "type_id")
    private RoomType roomType;

    @Column(name = "price")
    private int price;
```

```java
@Entity
@Table(name = "artist")
@NoArgsConstructor
@AllArgsConstructor
@ToString
@Data
public class Artist {
    @Id
    @GeneratedValue(strategy = Generat
    @Column(name = "id")
    private long id;

    @Column(name = "name")
    private @NonNull String name;

    @Column(name = "genre")
    private String genre;

    @Column(name = "phone")
    private @NonNull String phone;

    @Column(name = "email")
    private String email;
}
```

Схемы (архитектура, БД)

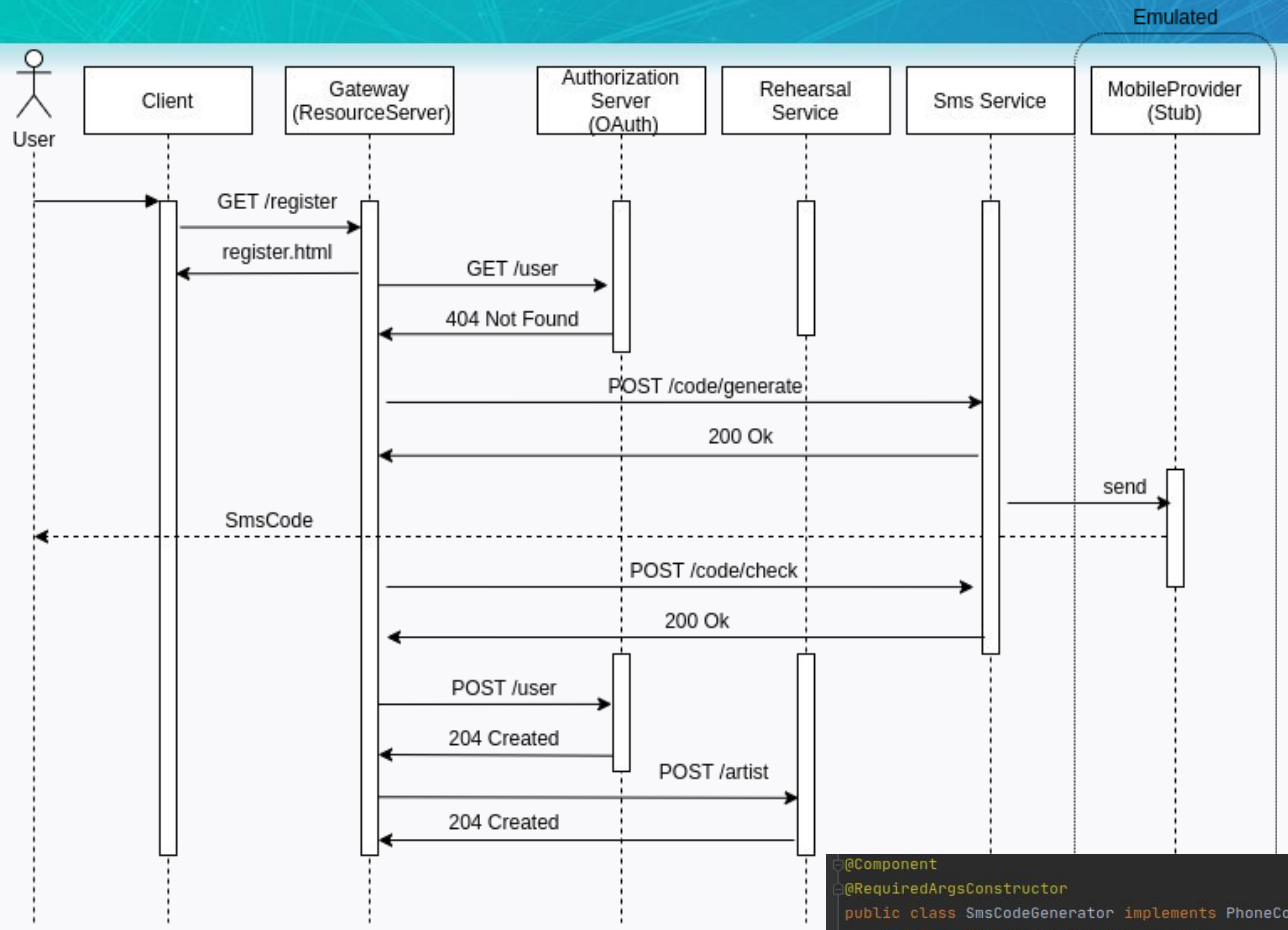



```
@PostMapping("/userLogin")
@ResponseBody
public ResponseEntity<AccessToken> login(UserLogin user) {
    logger.info("Trying to login {}", user);

    return
        userService
            .accessToken(user.getPhone(), user.getPassword())
            .map(
                token -> {
                    logger.info("Successfully got access token for {}", user.getPhone());

                    return ok(token);
                }
            )
            .orElseGet(
                () -> {
                    logger.warn("No access token for {}", user.getPhone());

                    return notFound().build();
```

```
$('#confirm-rehearsal').click(
  function () {
    $.ajax({
      url: `${path}/rehearsal`,
      type: "post",
      beforeSend: function (xhr) {
        xhr.setRequestHeader('Authorization', 'Bearer ' + token);
      },
      contentType: "application/json",
      data: JSON.stringify(rehearsal),
```

```
public class CustomTokenEnhancer implements TokenEnhancer {
    public static final String ID_KEY = "id";
    public static final String NAME_KEY = "name";
    public static final String PHONE_KEY = "phone";
    public static final String EMAIL_KEY = "email";
    public static final String ROLE_KEY = "role";

    @Override
    public OAuth2AccessToken enhance(OAuth2AccessToken accessToken, OAuth2Authentication authentication) {
        var user = (RehearsalBaseUser) authentication.getPrincipal();

        ((DefaultOAuth2AccessToken) accessToken).setAdditionalInformation(
            Map.of(
                ID_KEY, user.getId(),
                NAME_KEY, user.getUsername(),
                PHONE_KEY, user.getPhone(),
                EMAIL_KEY, user.getEmail(),
                ROLE_KEY,
                user.getAuthorities().stream().findFirst().map(GrantedAuthority::getAuthority).orElse("")
            )
        );

        return accessToken;
    }
}
```

Схемы (архитектура, БД)




```java
@GetMapping("/currentUser")
public ResponseEntity<User> currentUser(Authentication authentication) {
    return ResponseEntity.of(userService.authorizedUser(authentication));
}

@PostMapping("/register")
public ResponseEntity<Artist> register(@RequestBody ArtistUser artist) {
    logger.info("Registering new artist {}", artist);

    var result =
        userService
            .register(user(artist))
            .flatMap(authServerUser -> {
                logger.info(
                    "{} successfully registered in auth server. Registering as rehearsal service artist",
                    artist
                );

                return
                    artistService.create(
                        new Artist(artist.getName(), artist.getGenre(),artist.getPhone(), artist.getEmail())
                    );
            });

    return ResponseEntity.of(result);
}
```

```java
@Component
@RequiredArgsConstructor
public class SmsCodeGenerator implements PhoneCodeGenera
    private final SmsCodeConfig config;

    @Override
    public SmsCode generate(String phone) {
        return
            new SmsCode(
                phone,
                String.valueOf(new Random().nextInt(8999
                LocalDateTime.now().plusSeconds(config.g
            );
    }
}
```

```java
@PostMapping("code/generate")
public ResponseEntity<?> generate(@RequestBody PhoneDto phone) {
    var codeGenerationAvailability = provider.run(phone.getNumber());

    return
        codeGenerationAvailability.isAvailable()
            ? ok().build()
            : badRequest().build()
        ;
}

@PostMapping("code/check")
public ResponseEntity<SmsCodeStatus> checkCode(@RequestBody SmsCodeDto smsCode) {
    var result =  service.checkCode(new SmsCode(smsCode.getPhone(), smsCode.getCode()));

    return result.isOk() ? ok(result) : badRequest().build();
}
```

Вывод и планы по развитию

Написание проекта:

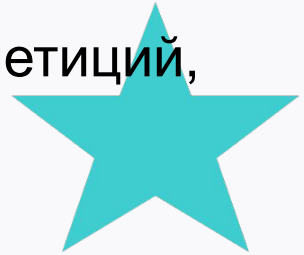
1. Нельзя закончить, можно прекратить)
2. Всегда есть место новому - oauth + jwt (Keycloak), frontend, docker
3. Новое, как и фронтенд, требует времени :)
4. Архитектура и функционал заложены, можно показывать и развивать

Spring:

1. Удобно и быстро разворачивать web-проекты, многое работает “из коробки”
2. Особенно, что касается микросервисов

Планы:

1. Прикрутить ELK
2. Добавить функционал администратора, аренды оборудования, горячих репетиций, динамическое ценообразование
3. Реализовать фронтенд на современном стеке



Спасибо за внимание!

Белов Василий

Backend-разработчик

belovi-tech@mail.ru