

ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Факультет компьютерных наук
Департамент программной инженерии

СОГЛАСОВАНО

УТВЕРЖДАЮ

Старший преподаватель департамента
программной инженерии факультета
компьютерных наук

Академический руководитель
образовательной программы
«Программная инженерия» профессор
департамента программной инженерии,
канд. техн. наук

_____ А. В. Поповкин
«_____» _____ 2021 г.

_____ В. В. Шилов
«_____» _____ 2021 г.

ПРИЛОЖЕНИЕ ДЛЯ СОВМЕСТНОГО ПРОСМОТРА
ФИЛЬМОВ

Текст программы

Лист УТВЕРЖДЕНИЯ

RU.17701729.02.07-01 12 01-1-ЛУ

Исполнитель: Студент группы БПИ-194
_____ В. А. Анненков
«_____» _____ 2021 г.

УТВЕРЖДЁН
RU.17701729.02.07-01 12 01-1-ЛУ

ПРИЛОЖЕНИЕ ДЛЯ СОВМЕСТНОГО ПРОСМОТРА
ФИЛЬМОВ

Текст программы

RU.17701729.02.07-01 12 01-1

Листов 39

Инв. № подл	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

Содержание

1	Текст программы сервиса MainService	4
1.1	MainServiceApplication.kt	4
1.2	MainController.kt	4
1.3	WebsocketController.kt	5
1.4	RoomService.kt	7
1.5	UserService.kt	10
1.6	WebsocketService.kt	10
1.7	RoomRepository.kt	11
1.8	UserRepository.kt	11
1.9	WebsocketEventListener.kt	11
1.10	Action.kt	12
1.11	Room.kt	12
1.12	User.kt	13
1.13	RoomActionReadyRequest.kt	13
1.14	RoomActionRequest.kt	13
1.15	RoomChatMessageRequest.kt	14
1.16	RoomReactionRequest.kt	14
1.17	Response.kt	14
1.18	RoomActionResponse.kt	14
1.19	RoomChatMessageResponse.kt	15
1.20	RoomJoinResponse.kt	15
1.21	RoomLeftResponse.kt	15
1.22	RoomReactionResponse.kt	15
1.23	WebsocketEnums.kt	16
1.24	ActionsStore.kt	17
1.25	AppConfig.kt	17
1.26	JWTTokenStoreConfig.kt	18
1.27	ResourceServerConfiguration.kt	18
1.28	SwaggerConfig.kt	19
1.29	WebSecurityConfig.kt	20
1.30	CustomHandshakeHandler.kt	21
1.31	StompPrincipal.kt	21
1.32	WebSocketConfig.kt	21
1.33	application.yml	22
1.34	V1__init.sql	23
2	Текст программы сервиса VideoService	24
2.1	MainServiceApplication.java	24
2.2	VideoController.kt	25
2.3	FfmpegManager.java	26
2.4	VideoResolution.kt	28
2.5	VideoService.kt	29
2.6	VideoUploadResponse.kt	32
2.7	MainClient.kt	32
2.8	AppConfig.java	32
2.9	JWTTokenStoreConfig.java	33
2.10	ResourceServerConfiguration.java	34

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.02.07-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

2.11	SwaggerConfig.kt	34
2.12	WebSecurityConfig.java	36
2.13	application.yml	36
3	Текст программы сервиса EurekaServer	37
3.1	application.yml	37
4	Текст программы сервиса GatewayServer	37
4.1	application.yml	37

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.02.07-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

1. Текст программы сервиса MainService

1.1. MainServiceApplication.kt

```
package tv.comnata.mainservice

import io.dekorate.kubernetes.annotation.ImagePullPolicy
import io.dekorate.kubernetes.annotation.KubernetesApplication
import io.dekorate.kubernetes.annotation.Probe
import io.dekorate.kubernetes.annotation.ServiceType
import org.springframework.boot.autoconfigure.SpringBootApplication
import org.springframework.boot.runApplication

@SpringBootApplication
@KubernetesApplication(
    livenessProbe = Probe(httpActionPath = "/"),
    readinessProbe = Probe(httpActionPath = "/"),
    serviceType = ServiceType.NodePort,
    imagePullPolicy = ImagePullPolicy.Always
)
class MainServiceApplication

fun main(args: Array<String>) {
    runApplication<MainServiceApplication>(*args)
}
```

1.2. MainController.kt

```
package tv.comnata.mainservice.controllers

import org.slf4j .LoggerFactory
import org.springframework.beans.factory.annotation.Autowired
import org.springframework.web.bind.annotation.*
import tv.comnata.mainservice.services.RoomService

@RestController
class MainController(
    @Autowired private val roomService: RoomService,
) {
    @GetMapping("/isRoomExist")
    fun checkIsRoomExist(@RequestParam roomName: String): Boolean {
        return roomService.checkIsRoomExist(roomName)
    }

    @RequestMapping("/createVideo", method = [RequestMethod.PUT])
    fun createVideo(@RequestParam videoUuid: String) {
        roomService.createRoom(videoUuid)
    }
}
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.02.07-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

@PostMapping("/setVideoProgress")
fun setVideoProgress(@RequestParam videoUuid: String, @RequestParam videoProgress: Int)
{
    // roomService.setVideoProgress(videoUuid, videoProgress)
}

companion object {
    private val logger = LoggerFactory.getLogger(MainController::class.java)
}
}

```

1.3. WebsocketController.kt

```
package tv.comnata.mainservice.controllers
```

```

import org.slf4j .LoggerFactory
import org.springframework.beans.factory.annotation.Autowired
import org.springframework.messaging.handler.annotation.DestinationVariable
import org.springframework.messaging.handler.annotation.MessageMapping
import org.springframework.messaging.handler.annotation.Payload
import org.springframework.stereotype.Controller
import org.springframework.web.bind.annotation.RequestMapping
import org.springframework.web.bind.annotation.RequestMethod
import tv.comnata.mainservice.entities.websocket.getActionType
import tv.comnata.mainservice.entities.websocket.getReaction
import tv.comnata.mainservice.entities.websocket.requests.RoomActionReadyRequest
import tv.comnata.mainservice.entities.websocket.requests.RoomActionRequest
import tv.comnata.mainservice.entities.websocket.requests.RoomChatMessageRequest
import tv.comnata.mainservice.entities.websocket.requests.RoomReactionRequest
import tv.comnata.mainservice.services.RoomService
import java.security .Principal

```

```

@Controller
class WebsocketController(
    @Autowired
    private val roomService: RoomService,
) {
    @RequestMapping(URL_ROOM_JOIN, method = [RequestMethod.POST])
    @MessageMapping(URL_ROOM_JOIN)
    fun processRoomJoin(
        principal: Principal,
        @DestinationVariable roomId: String,
    ) {
        logger.info("JOIN")
        roomService.processVideoJoin(principal.name, roomId)
    }
}

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.02.07-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

@RequestMapping(URL_ROOM_VIDEO_ACTION, method = [RequestMethod.POST])
@MessageMapping(URL_ROOM_VIDEO_ACTION)
fun processRoomVideoAction(
    principal: Principal,
    @DestinationVariable roomId: String,
    @Payload request: RoomActionRequest
) {
    logger.info("VIDEO ACTION \t ${request.type}")
    roomService.processRoomVideoAction(
        principal.name,
        roomId,
        request.seekTime!!,
        request.type!!.getActionType()
    )
}

@RequestMapping(URL_ROOM_VIDEO_ACTION_READY, method = [RequestMethod.
    POST])
@MessageMapping(URL_ROOM_VIDEO_ACTION_READY)
fun processRoomVideoActionReady(
    principal: Principal,
    @DestinationVariable roomId: String,
    @Payload request: RoomActionReadyRequest,
) {
    logger.info("READY \t ${principal.name}")
    roomService.processRoomVideoActionReady(principal.name, roomId, request.actionId!!)
}

@RequestMapping(URL_ROOM_CHAT_MESSAGE, method = [RequestMethod.POST])
@MessageMapping(URL_ROOM_CHAT_MESSAGE)
fun processRoomChatMessage(
    principal: Principal,
    @DestinationVariable roomId: String,
    @Payload request: RoomChatMessageRequest
) {
    logger.info("CHAT MESSAGE")
    roomService.processRoomChatMessage(principal.name, roomId, request.text!!)
}

@RequestMapping(URL_ROOM_REACTION, method = [RequestMethod.POST])
@MessageMapping(URL_ROOM_REACTION)
fun processRoomReaction(
    principal: Principal,
    @DestinationVariable roomId: String,
    @Payload request: RoomReactionRequest
) {
    logger.info("REACTION")
    roomService.processRoomReaction(principal.name, roomId, request.reaction!!.getReaction

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.02.07-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

        ()
    }

    companion object {
        private val logger = LoggerFactory.getLogger(WebsocketController::class.java)

        const val URL_ROOM_JOIN = "/room/{roomId}/join"
        const val URL_ROOM_VIDEO_ACTION = "/room/{roomId}/videoAction"
        const val URL_ROOM_VIDEO_ACTION_READY = "/room/{roomId}/videoActionReady"
        const val URL_ROOM_CHAT_MESSAGE = "/room/{roomId}/chatMessage"
        const val URL_ROOM_REACTION = "/room/{roomId}/reaction"
    }
}

```

1.4. RoomService.kt

```

package tv.comnata.mainservice.services

import org.springframework.beans.factory.annotation.Autowired
import org.springframework.stereotype.Service
import org.springframework.transaction.annotation.Transactional
import tv.comnata.mainservice.entities.Action
import tv.comnata.mainservice.entities.Room
import tv.comnata.mainservice.entities.User
import tv.comnata.mainservice.entities.websocket.ActionStep
import tv.comnata.mainservice.entities.websocket.ActionType
import tv.comnata.mainservice.entities.websocket.Reaction
import tv.comnata.mainservice.entities.websocket.getActionType
import tv.comnata.mainservice.entities.websocket.responses.*
import tv.comnata.mainservice.repositories.RoomRepository
import tv.comnata.mainservice.repositories.UserRepository
import tv.comnata.mainservice.stores.ActionsStore
import java.time.LocalDateTime

@Service
class RoomService(
    @Autowired private val websocketService: WebsocketService,
    @Autowired private val userRepository: UserRepository,
    @Autowired private val roomRepository: RoomRepository,
    @Autowired private val actionsStore: ActionsStore,
) {
    fun checkIsRoomExist(roomName: String): Boolean {
        return roomRepository.findRoomByName(roomName) != null
    }

    fun createRoom(roomName: String) {
        val room = Room(

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.02.07-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата


```

        roomName,
        LocalDateTime.now(),
    )

    roomRepository.saveAndFlush(room)
}

fun processVideoJoin(userId: String, roomId: String) {
    val room = roomRepository.findRoomByName(roomId)
    val user = User(userId, room!!)
    userRepository.save(user)
    val users = userRepository.findAllByRoomName(roomId).map { it.username }

    websocketService.send(
        URL_ROOM_JOINS.format(roomId),
        RoomJoinResponse(userId, users, LocalDateTime.now())
    )
}

fun processVideoLeft(userId: String) {
    val user = userRepository.findUserByUsername(userId)
    val room = roomRepository.findRoomByName(user.room.name)
    userRepository.delete(user)

    val users = userRepository.findAllByRoomName(room!!.name).map { it.username }
    websocketService.send(
        URL_ROOM_LEFTS.format(room.name),
        RoomLeftResponse(userId, users, LocalDateTime.now())
    )
}

@Transactional
fun processRoomVideoAction(userId: String, roomId: String, seekTime: Double, type:
    ActionType) {
    if (type == ActionType.SEEK) {
        processRoomVideoActionSeek(userId, roomId, seekTime, type)
    } else {
        websocketService.send(
            URL_ROOM_ACTIONS.format(roomId),
            RoomActionResponse(-1, userId, seekTime, type, ActionStep.READY,
                LocalDateTime.now())
        )
    }
}

@Transactional
fun processRoomVideoActionSeek(userId: String, roomId: String, seekTime: Double, type:
    ActionType) {

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.02.07-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

val room = roomRepository.findRoomByName(roomId)
val action = Action(actionsStore.index, type.name, seekTime, userId)
action.addUsers(room!!.users.map { it.username }.toMutableSet())
actionsStore.waitingActions[action.id] = action

websocketService.send(
    URL_ROOM_ACTIONS.format(roomId),
    RoomActionResponse(action.id, userId, seekTime, type, ActionStep.CHECK,
        LocalDateTime.now())
)
}

fun processRoomVideoActionReady(userId: String, roomId: String, actionId: Long) {
    val action = actionsStore.waitingActions[actionId]
    action!!.deleteUser(userId)

    if (action.users.isEmpty()) {
        websocketService.send(
            URL_ROOM_ACTIONS.format(roomId),
            RoomActionResponse(
                actionId,
                userId,
                action.seekTime,
                action.type.getActionType(),
                ActionStep.READY,
                LocalDateTime.now()
            )
        )
        actionsStore.waitingActions.remove(actionId)
    }
}

fun processRoomChatMessage(userId: String, roomId: String, text: String) {
    websocketService.send(
        URL_ROOM_CHAT_MESSAGES.format(roomId),
        RoomChatMessageResponse(userId, text, LocalDateTime.now())
    )
}

fun processRoomReaction(userId: String, roomId: String, reaction: Reaction) {
    websocketService.send(
        URL_ROOM_REACTIONS.format(roomId),
        RoomReactionResponse(userId, reaction, LocalDateTime.now())
    )
}

companion object {
    private const val URL_ROOM_JOINS = "/topic/room/%s/joins"

```

ИЗМ.	Лист	№ докум.	Подп.	Дата
RU.17701729.02.07-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```
        private const val URL_ROOM_LEFTS = "/topic/room/%s/lefts"
        private const val URL_ROOM_ACTIONS = "/topic/room/%s/videoActions"
        private const val URL_ROOM_CHAT_MESSAGES = "/topic/room/%s/chatMessages"
        private const val URL_ROOM_REACTIONS = "/topic/room/%s/reactions"
    }
}
```

1.5. UserService.kt

```
package tv.comnata.mainservice.services

import org.springframework.beans.factory.annotation.Autowired
import org.springframework.stereotype.Service
import tv.comnata.mainservice.entities.User
import tv.comnata.mainservice.repositories.UserRepository

@Service
class UserService(
    @Autowired
    private val repository: UserRepository
) {
    fun getUser(): User {
        val user = repository.findUserByUsername("Vakosta")
        return user
    }
}
```

1.6. WebsocketService.kt

```
package tv.comnata.mainservice.services

import org.springframework.beans.factory.annotation.Autowired
import org.springframework.messaging.simp.SimpMessagingTemplate
import org.springframework.messaging.simp.user.SimpUserRegistry
import org.springframework.stereotype.Service

@Service
class WebsocketService(
    @Autowired private val messagingTemplate: SimpMessagingTemplate,
    @Autowired private val simpUserRegistry: SimpUserRegistry,
) {
    fun send(url: String, obj: Any) {
        messagingTemplate.convertAndSend(url, obj)
    }

    fun sendToUser(url: String, user: String, obj: Any) {
        messagingTemplate.convertAndSendToUser(user, url, obj)
    }
}
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.02.07-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```
fun getNumberOfSessions(): Int {  
    return simpUserRegistry.userCount  
}  
}
```

1.7. RoomRepository.kt

```
package tv.comnata.mainservice.repositories  
  
import org.springframework.data.jpa.repository.JpaRepository  
import tv.comnata.mainservice.entities.Room  
  
interface RoomRepository : JpaRepository<Room, Long> {  
    fun saveAndFlush(room: Room)  
    fun findRoomByName(name: String): Room?  
}
```

1.8. UserRepository.kt

```
package tv.comnata.mainservice.repositories  
  
import org.springframework.data.jpa.repository.JpaRepository  
import tv.comnata.mainservice.entities.User  
  
interface UserRepository : JpaRepository<User, Long> {  
    fun findUserByUsername(name: String): User  
    fun findAllByRoomName(roomName: String): List<User>  
}
```

1.9. WebSocketEventListener.kt

```
package tv.comnata.mainservice.eventlisteners  
  
import org.slf4j.LoggerFactory  
import org.springframework.beans.factory.annotation.Autowired  
import org.springframework.context.event.EventListener  
import org.springframework.stereotype.Component  
import org.springframework.web.socket.messaging.SessionConnectEvent  
import org.springframework.web.socket.messaging.SessionDisconnectEvent  
import tv.comnata.mainservice.services.RoomService  
  
@Component  
class WebSocketEventListener(  
    @Autowired  
    private val roomService: RoomService,  
) {  
    @EventListener
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.02.07-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```
fun handleSessionConnect(event: SessionConnectEvent) {
    logger.info("Connected \t ${event.user!!.name}")
}

@EventListener
fun handleSessionDisconnect(event: SessionDisconnectEvent) {
    logger.info("Disconnected \t ${event.user!!.name}")
    roomService.processVideoLeft(event.user!!.name)
}

companion object {
    private val logger = LoggerFactory.getLogger(WebsocketEventListener::class.java)
}
}
```

1.10. Action.kt

```
package tv.comnata.mainservice.entities

class Action(
    var id: Long,

    var type: String,

    var seekTime: Double,

    var author: String,

    var users: MutableSet<String> = HashSet(),
) {
    fun addUser(user: String) {
        users.add(user)
    }

    fun addUsers(newUsers: Set<String>) {
        users.addAll(newUsers)
    }

    fun deleteUser(user: String) {
        users.remove(user)
    }

    fun deleteAllUsers() {
        users.clear()
    }
}
```

1.11. Room.kt

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.02.07-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```
package tv.comnata.mainservice.entities

import java.time.LocalDateTime
import javax.persistence.*

@Entity
@Table(name = "room")
class Room(
    var name: String,

    var creationDate: LocalDateTime,

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    var id: Long? = null,

    @OneToMany(mappedBy = "room", fetch = FetchType.LAZY)
    var users: Set<User> = HashSet(),
)
```

1.12. User.kt

```
package tv.comnata.mainservice.entities

import javax.persistence.*

@Entity
@Table(name = "app_user")
class User(
    var username: String,

    @ManyToOne(fetch = FetchType.LAZY)
    var room: Room,

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    var id: Long? = null,
)
```

1.13. RoomActionReadyRequest.kt

```
package tv.comnata.mainservice.entities.websocket.requests

class RoomActionReadyRequest {
    val actionId: Long? = null
}
```

1.14. RoomActionRequest.kt

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.02.07-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```
package tv.comnata.mainservice.entities.websocket.requests
```

```
class RoomActionRequest {  
    val seekTime: Double? = null  
    val type: String? = null  
}
```

1.15. RoomChatMessageRequest.kt

```
package tv.comnata.mainservice.entities.websocket.requests
```

```
class RoomChatMessageRequest {  
    val text: String? = null  
}
```

1.16. RoomReactionRequest.kt

```
package tv.comnata.mainservice.entities.websocket.requests
```

```
class RoomReactionRequest {  
    val reaction: String? = null  
}
```

1.17. Response.kt

```
package tv.comnata.mainservice.entities.websocket.responses
```

```
abstract class Response(  
    val notificationType: NotificationType,  
) {  
    enum class NotificationType {  
        JOIN,  
        LEFT,  
        VIDEO_ACTION,  
        CHAT_MESSAGE,  
        REACTION,  
    }  
}
```

1.18. RoomActionResponse.kt

```
package tv.comnata.mainservice.entities.websocket.responses
```

```
import tv.comnata.mainservice.entities.websocket.ActionStep  
import tv.comnata.mainservice.entities.websocket.ActionType  
import java.time.LocalDateTime
```

```
data class RoomActionResponse(  

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.02.07-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```
    val actionId: Long,  
    val author: String,  
    val seekTime: Double,  
    val type: ActionType,  
    val step: ActionStep,  
    val actionTime: LocalDateTime,  
    ) : Response(NotificationType.VIDEO_ACTION)
```

1.19. RoomChatMessageResponse.kt

```
package tv.comnata.mainservice.entities.websocket.responses  
  
import java.time.LocalDateTime  
  
data class RoomChatMessageResponse(  
    val userId: String,  
    val text: String,  
    val dateTime: LocalDateTime,  
    ) : Response(NotificationType.CHAT_MESSAGE)
```

1.20. RoomJoinResponse.kt

```
package tv.comnata.mainservice.entities.websocket.responses  
  
import java.time.LocalDateTime  
  
data class RoomJoinResponse(  
    val newUserId: String,  
    val allUserIds: List<String>,  
    val dateTime: LocalDateTime,  
    ) : Response(NotificationType.JOIN)
```

1.21. RoomLeftResponse.kt

```
package tv.comnata.mainservice.entities.websocket.responses  
  
import java.time.LocalDateTime  
  
data class RoomLeftResponse(  
    val leftUserId: String,  
    val remainingUserIds: List<String>,  
    val dateTime: LocalDateTime  
    ) : Response(NotificationType.LEFT)
```

1.22. RoomReactionResponse.kt

```
package tv.comnata.mainservice.entities.websocket.responses
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.02.07-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата


```
import tv.comnata.mainservice.entities.websocket.Reaction
import java.time.LocalDateTime
```

```
data class RoomReactionResponse(
    val author: String,
    val type: Reaction,
    val actionTime: LocalDateTime,
) : Response(NotificationType.REACTION)
```

1.23. WebsocketEnums.kt

```
package tv.comnata.mainservice.entities.websocket
```

```
enum class ActionType {
    RESUME,
    PAUSE,
    SEEK,
    ALL_CLIENTS_READY,
    UNDEFINED,
}
```

```
enum class ActionStep {
    CHECK,
    READY,
}
```

```
enum class Reaction {
    GOOD,
    OMG,
    ANGRY,
    UNDEFINED,
}
```

```
fun String.getActionType(): ActionType {
    return when (this.toUpperCase()) {
        "RESUME" -> ActionType.RESUME
        "PAUSE" -> ActionType.PAUSE
        "SEEK" -> ActionType.SEEK
        else -> ActionType.UNDEFINED
    }
}
```

```
fun String.getReaction(): Reaction {
    return when (this.toUpperCase()) {
        "GOOD" -> Reaction.GOOD
        "OMG" -> Reaction.OMG
        "ANGRY" -> Reaction.ANGRY
        else -> Reaction.UNDEFINED
    }
}
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.02.07-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```
}  
}
```

1.24. ActionsStore.kt

```
package tv.comnata.mainservice.stores  
  
import org.slf4j .LoggerFactory  
import org.springframework.stereotype.Component  
import tv.comnata.mainservice.entities .Action  
import java. util .concurrent.ConcurrentHashMap  
  
@Component  
class ActionsStore {  
    @Volatile  
    var index = 0L  
    get() {  
        logger .info("New index — $field ")  
        field ++  
        return field  
    }  
  
    val waitingActions = ConcurrentHashMap<Long, Action>()  
  
    companion object {  
        private val logger = LoggerFactory.getLogger(ActionsStore::class.java)  
    }  
}
```

1.25. AppConfig.kt

```
package tv.comnata.mainservice.configs  
  
import org.springframework.boot.web.servlet.MultipartConfigFactory  
import org.springframework.context.annotation.Bean  
import org.springframework.context.annotation.ComponentScan  
import org.springframework.context.annotation.Configuration  
import org.springframework.util.unit .DataSize  
import javax. servlet .MultipartConfigElement  
  
@Configuration  
@ComponentScan("tv.comnata.mainservice")  
class AppConfig {  
    @Bean  
    fun multipartConfigElement(): MultipartConfigElement {  
        val factory = MultipartConfigFactory()  
        factory .setMaxFileSize(DataSize.ofGigabytes(7))  
        factory .setMaxRequestSize(DataSize.ofGigabytes(7))  
    }  
}
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.02.07-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```
        return factory.createMultipartConfig()
    }
}
```

1.26. JWTTokenStoreConfig.kt

```
package tv.comnata.mainservice.configs
```

```
import org.springframework.context.annotation.Bean
import org.springframework.context.annotation.Configuration
import org.springframework.context.annotation.Primary
import org.springframework.security.oauth2.provider.token.DefaultTokenServices
import org.springframework.security.oauth2.provider.token.TokenStore
import org.springframework.security.oauth2.provider.token.store.JwtAccessTokenConverter
import org.springframework.security.oauth2.provider.token.store.JwtTokenStore
```

```
@Configuration
```

```
class JWTTokenStoreConfig {
```

```
    @Bean
```

```
    fun tokenStore(): TokenStore {
        return JwtTokenStore(jwtAccessTokenConverter())
    }
```

```
    @Bean
```

```
    @Primary
```

```
    fun tokenServices(): DefaultTokenServices {
        val defaultTokenServices = DefaultTokenServices()
        defaultTokenServices.setTokenStore(tokenStore())
        defaultTokenServices.setSupportRefreshToken(true)
        return defaultTokenServices
    }
```

```
    @Bean
```

```
    fun jwtAccessTokenConverter(): JwtAccessTokenConverter {
        val converter = JwtAccessTokenConverter()
        converter.setSigningKey("helloworld")
        return converter
    }
}
```

```
}
```

1.27. ResourceServerConfiguration.kt

```
package tv.comnata.mainservice.configs
```

```
import org.springframework.beans.factory.annotation.Qualifier
import org.springframework.context.annotation.Bean
import org.springframework.context.annotation.Configuration
import org.springframework.security.config.annotation.web.builders.HttpSecurity
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.02.07-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```
import org.springframework.security.oauth2.client .OAuth2ClientContext
import org.springframework.security.oauth2.client .OAuth2RestTemplate
import org.springframework.security.oauth2.client .resource .OAuth2ProtectedResourceDetails
import org.springframework.security.oauth2.config.annotation.web.configuration.
    ResourceServerConfigurerAdapter

@Configuration
class ResourceServerConfiguration : ResourceServerConfigurerAdapter() {
    @Bean
    fun oauth2RestTemplate(
        @Qualifier("oauth2ClientContext") oauth2ClientContext: OAuth2ClientContext?,
        details : OAuth2ProtectedResourceDetails?
    ): OAuth2RestTemplate {
        return OAuth2RestTemplate(details, oauth2ClientContext)
    }

    @Throws(Exception::class)
    override fun configure(http: HttpSecurity) {
        http.authorizeRequests()

            .anyRequest().permitAll()
    }
}
```

1.28. SwaggerConfig.kt

```
package tv.comnata.mainservice.configs

import org.springframework.context.annotation.Bean
import org.springframework.context.annotation.Configuration
import org.springframework.web.servlet.config.annotation.CorsRegistry
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer
import springfox.documentation.builders.PathSelectors
import springfox.documentation.builders.RequestHandlerSelectors
import springfox.documentation.service.*
import springfox.documentation.spi.DocumentationType
import springfox.documentation.spring.web.plugins.Docket

@Configuration
class SwaggerConfig : WebMvcConfigurer {
    @Bean
    fun api(): Docket? {
        val contact = Contact(
            "Vladislav Annenkov",
            "https://t.me/Vakosta",
            "v.a.annenkov@ya.ru"
        )
    }
}
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.02.07-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

val vext: List<VendorExtension<*>> = ArrayList()
val apiInfo = ApiInfo(
    "Comnata Main API",
    "API for main features of the Comnata.",
    "1.0.0",
    "https://something.com",
    contact,
    "MIT",
    "https://something.com",
    vext
)

return Docket(DocumentationType.SWAGGER_2)
    .apiInfo(apiInfo)
    .forCodeGeneration(true)
    .securitySchemes(listOf(apiKey()) as List<SecurityScheme>?)
    .select ()
    .apis(RequestHandlerSelectors.basePackage("tv.comnata.mainservice.controllers"))
    .paths(PathSelectors.any())
    .build()
}

override fun addResourceHandlers(registry: org.springframework.web.servlet.config.annotation
.ResourceHandlerRegistry) {
    registry .addResourceHandler("swagger-ui.html")
        .addResourceLocations("classpath:/META-INF/resources/")
    registry .addResourceHandler("/webjars/**")
        .addResourceLocations("classpath:/META-INF/resources/webjars/")
}

private fun apiKey(): ApiKey {
    return ApiKey("authkey", "Authorization", "header")
}

override fun addCorsMappings(registry: CorsRegistry) {
    registry .addMapping("/**")
}
}

```

1.29. WebSecurityConfig.kt

```
package tv.comnata.mainservice.configs
```

```

import org.springframework.security.config.annotation.web.builders.HttpSecurity
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity
import org.springframework.security.config.annotation.web.configuration.
    WebSecurityConfigurerAdapter

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.02.07-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```
@EnableWebSecurity
class WebSecurityConfig : WebSecurityConfigurerAdapter() {
    @Throws(Exception::class)
    override fun configure(http: HttpSecurity) {
        // http.cors()
        http.csrf().disable()
    }
}
```

1.30. CustomHandshakeHandler.kt

```
package tv.comnata.mainservice.configs.websocket

import org.springframework.http.server.ServerHttpRequest
import org.springframework.web.socket.WebSocketHandler
import org.springframework.web.socket.server.support.DefaultHandshakeHandler
import java.security.Principal
import java.util.*

class CustomHandshakeHandler : DefaultHandshakeHandler() {
    override fun determineUser(
        request: ServerHttpRequest,
        wsHandler: WebSocketHandler,
        attributes: MutableMap<String, Any>
    ): Principal {
        return StompPrincipal(UUID.randomUUID().toString())
    }
}
```

1.31. StompPrincipal.kt

```
package tv.comnata.mainservice.configs.websocket

import java.security.Principal

class StompPrincipal(private val name: String) : Principal {
    override fun getName(): String {
        return name
    }
}
```

1.32. WebSocketConfig.kt

```
package tv.comnata.mainservice.configs.websocket

import com.fasterxml.jackson.databind.ObjectMapper
import org.springframework.context.annotation.Configuration
import org.springframework.messaging.converter.DefaultContentTypeResolver
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.02.07-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

import org.springframework.messaging.converter.MappingJackson2MessageConverter
import org.springframework.messaging.converter.MessageConverter
import org.springframework.messaging.simp.config.MessageBrokerRegistry
import org.springframework.util.MimeTypeUtils
import org.springframework.web.socket.config.annotation.EnableWebSocketMessageBroker
import org.springframework.web.socket.config.annotation.StompEndpointRegistry
import org.springframework.web.socket.config.annotation.WebSocketMessageBrokerConfigurer

@Configuration
@EnableWebSocketMessageBroker
class WebSocketConfig : WebSocketMessageBrokerConfigurer {
    override fun configureMessageBroker(config: MessageBrokerRegistry) {
        config.enableSimpleBroker("/topic")
        config.setUserDestinationPrefix("/user")
        config.setApplicationDestinationPrefixes("/app")
    }

    override fun registerStompEndpoints(registry: StompEndpointRegistry) {
        registry
            .addEndpoint("/ws")
            .setAllowedOrigins("*")
            .setHandshakeHandler(CustomHandshakeHandler())
            .withSockJS()
    }

    override fun configureMessageConverters(messageConverters: MutableList<MessageConverter
?>): Boolean {
        val resolver = DefaultContentTypeResolver()
        resolver.defaultMimeType = MimeTypeUtils.APPLICATION_JSON
        val converter = MappingJackson2MessageConverter()
        converter.objectMapper = ObjectMapper()
        converter.contentTypeResolver = resolver
        messageConverters.add(converter)
        return false
    }
}

```

1.33. application.yml

```

spring:
  application:
    name: main-service
  datasource:
    url: jdbc:postgresql://ec2-54-247-158-179.eu-west-1.compute.amazonaws.com:5432/
      ddir5jgg0qg2k
    username: crhxbecsivpbwv
    password: 2a66eec5f7d9756b445b5113fca7eea05ec321160b627d37a8e51b5cae598dd5 # Don't
      worry about this.

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.02.07-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

    driver-class-name: org.postgresql.Driver
jpa:
    database: postgresql
    database-platform: org.hibernate.dialect.PostgreSQL10Dialect
flyway:
    url: postgres://crhxbecsivpbwv:2
        a66eec5f7d9756b445b5113fca7eea05ec321160b627d37a8e51b5cae598dd5@ec2
        -54-247-158-179.eu-west-1.compute.amazonaws.com:5432/ddisr5jgg0qg2k
    user: crhxbecsivpbwv
    password: 2a66eec5f7d9756b445b5113fca7eea05ec321160b627d37a8e51b5cae598dd5
cloud:
    kubernetes:
        enabled: false

server:
    port: 8192
    servlet:
        context-path: /main

eureka:
    client:
        service-url:
            defaultZone: ${EUREKA_URI:http://localhost:8761/eureka}
        fetchRegistry: true
        enabled: true

security:
    oauth2:
        resource:
            userInfoUri: ${AUTH_URI:http://localhost:8880/auth/user}
        client:
            client-id: flametoken
            client-secret: thisissecret

```

1.34. V1__init.sql

```
/* ===== ROOMS ===== */
```

```

CREATE TABLE room
(
    id          BIGSERIAL NOT NULL,
    name        VARCHAR(10) NOT NULL,
    creation_date TIMESTAMP NOT NULL DEFAULT NOW(),

    PRIMARY KEY (id)
);

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.02.07-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата


```
/* ===== USERS ===== */  
  
CREATE TABLE app_user  
(  
    id          BIGSERIAL NOT NULL,  
    username VARCHAR(50) NOT NULL,  
    room_id BIGSERIAL NOT NULL,  
  
    PRIMARY KEY (id),  
  
    FOREIGN KEY (room_id) REFERENCES room (id)  
        ON DELETE CASCADE  
        ON UPDATE CASCADE  
);
```

2. Текст программы сервиса VideoService

2.1. MainServiceApplication.java

```
package tv.comnata.videoservice;  
  
import io.dekorate.kubernetes.annotation.ImagePullPolicy;  
import io.dekorate.kubernetes.annotation.KubernetesApplication;  
import io.dekorate.kubernetes.annotation.Probe;  
import io.dekorate.kubernetes.annotation.ServiceType;  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
import org.springframework.cloud.client.discovery.EnableDiscoveryClient;  
import org.springframework.cloud.openfeign.EnableFeignClients;  
import org.springframework.security.oauth2.config.annotation.web.configuration.  
    EnableResourceServer;  
  
@EnableFeignClients  
@EnableDiscoveryClient  
@EnableResourceServer  
@SpringBootApplication  
@KubernetesApplication(  
    livenessProbe = @Probe(httpActionPath = "/"),  
    readinessProbe = @Probe(httpActionPath = "/"),  
    serviceType = ServiceType.NodePort,  
    imagePullPolicy = ImagePullPolicy.IfNotPresent  
)  
public class VideoServiceApplication {  
    public static void main(String[] args) {  
        SpringApplication.run(VideoServiceApplication.class, args);  
    }  
}
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.02.07-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

2.2. VideoController.kt

```

package tv.comnata.videoservice.controllers

import org.slf4j.LoggerFactory
import org.springframework.beans.factory.annotation.Autowired
import org.springframework.core.io.FileSystemResource
import org.springframework.http.HttpHeaders
import org.springframework.http.HttpStatus
import org.springframework.http.ResponseEntity
import org.springframework.stereotype.Controller
import org.springframework.web.bind.annotation.*
import org.springframework.web.multipart.MultipartFile
import tv.comnata.videoservice.entities.VideoUploadResponse
import tv.comnata.videoservice.services.VideoService
import javax.servlet.http.HttpServletRequest
import javax.servlet.http.HttpServletResponse

@Controller
@CrossOrigin(origins = ["*"])
class VideoController(
    @Autowired
    private var videoService: VideoService,
) {
    @GetMapping(value = ["/getVideo/{video_id}/{file_name}"], produces = [MEDIA_TYPE])
    fun getBaseFile(
        response: HttpServletResponse,
        @PathVariable("video_id") videoId: String,
        @PathVariable("file_name") fileName: String,
    ): ResponseEntity<FileSystemResource> {
        logger.info("VIDEO $videoId \t BASE FILE $fileName")

        val headers = HttpHeaders()
        response.setHeader("Content-Disposition", String.format("inline; filename=%s", fileName))
        val path = "/tmp/videos/$videoId/$fileName"
        return ResponseEntity(FileSystemResource(path), headers, HttpStatus.OK)
    }

    @GetMapping(value = ["/getVideo/{video_id}/{resolution}/{file_name}"], produces = [
        MEDIA_TYPE])
    fun getVideoFile(
        response: HttpServletResponse,
        @PathVariable("video_id") videoId: String,
        @PathVariable("resolution") resolution: String,
        @PathVariable("file_name") fileName: String,
    ): ResponseEntity<FileSystemResource> {
        logger.info("VIDEO $videoId \t FILE $fileName \t RESOLUTION $resolution")
    }
}

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.02.07-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

    val headers = HttpHeaders()
    response.setHeader("Content-Disposition", String.format("inline; filename=%s", fileName
    ))
    val path = "/tmp/videos/$videoId/$resolution/$fileName"
    return ResponseEntity(FileSystemResource(path), headers, HttpStatus.OK)
}

@ResponseBody
@PostMapping("/upload")
fun uploadFile(
    request: HttpServletRequest,
    @RequestParam file: MultipartFile
): VideoUploadResponse {
    logger.info("UPLOAD NEW FILE")
    return videoService.saveVideo(file, "/tmp/videos/")
}

companion object {
    private val logger = LoggerFactory.getLogger(VideoController::class.java)

    const val MEDIA_TYPE = "application/x-mpegURL"
}
}

```

2.3. FfmpegManager.java

```

package tv.comnata.videoservice.services;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.Scanner;
import java.util.regex.Pattern;

public class FfmpegManager extends Thread {
    private static final VideoResolution[] AVAILABLE_RESOLUTIONS = {
        new VideoResolution(426, 240),
        new VideoResolution(640, 360),
        new VideoResolution(852, 480),
        new VideoResolution(1280, 720),
        new VideoResolution(1920, 1080),
    };

    private static final String COMMAND_HLS_BASE = "ffmpeg_-i_%s";
    private static final String COMMAND_HLS_ONE_RESOLUTION = "_-c:a_aac_-strict_
experimental_-c:v_libx264_" +
        "-s_%s_-aspect_16:9_-f_hls_-hls_list_size_0_-hls_time_10_-threads_0_%sp/video.

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.02.07-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

        m3u8";
private static final String COMMAND_RESOLUTION =
    "ffprobe_-v_error_-select_streams_v:0_-show_entries_stream=width,height_-of_csv=s
    =x:p=0_%s";

private final String path;
private final String fileName;
private final OnUpdateProgressListener listener;

public FfmpegManager(String path, String fileName, OnUpdateProgressListener listener) {
    this.path = path;
    this.fileName = fileName;
    this.listener = listener;
}

VideoResolution getVideoResolution() throws IOException {
    Process process = Runtime.getRuntime().exec(String.format(
        COMMAND_RESOLUTION, path + fileName));

    BufferedReader in = new BufferedReader(new InputStreamReader(process.getInputStream
        ()));
    String inputLine;
    while ((inputLine = in.readLine()) != null) {
        in.close();
        return new VideoResolution(inputLine);
    }

    throw new IOException();
}

private String getHlsCommand() throws IOException {
    VideoResolution resolution = getVideoResolution();
    StringBuilder command = new StringBuilder(String.format(COMMAND_HLS_BASE,
        path + fileName));

    for (VideoResolution availableResolution : AVAILABLE_RESOLUTIONS) {
        if (resolution.compareTo(availableResolution) < 0) {
            break;
        }

        command.append(String.format(COMMAND_HLS_ONE_RESOLUTION,
            availableResolution,
            path + availableResolution.getHeight()));
    }

    return command.toString();
}

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.02.07-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

@Override
public void run() {
    try {
        ProcessBuilder processBuilder = new ProcessBuilder(getHlsCommand().split("_"));
        final Process process = processBuilder.start();
        Scanner sc = new Scanner(process.getErrorStream());

        // Find duration
        Pattern durPattern = Pattern.compile("(?<=Duration:)[^,]*");
        String dur = sc.findWithinHorizon(durPattern, 0);
        if (dur == null) {
            throw new RuntimeException("Could_not_parse_duration.");
        }
        String[] hms = dur.split(":");
        double totalSecs = Integer.parseInt(hms[0]) * 3600
            + Integer.parseInt(hms[1]) * 60
            + Double.parseDouble(hms[2]);

        Pattern timePattern = Pattern.compile("(?<=time=)[\\d:.]*");
        String match;
        String[] matchSplit;
        while (!isInterrupted() && null != (match = sc.findWithinHorizon(timePattern, 0)))
        {
            matchSplit = match.split(":");
            double progress = (Integer.parseInt(matchSplit[0]) * 3600 +
                Integer.parseInt(matchSplit[1]) * 60 +
                Double.parseDouble(matchSplit[2])) / totalSecs;
            listener .onUpdatePercent(path, progress * 100);
        }
    } catch (IOException exception) {
        exception.printStackTrace();
    }
}

interface OnUpdateProgressListener {
    void onUpdatePercent(String videoUuid, double percent);
}
}

```

2.4. VideoResolution.kt

```
package tv.comnata.videoservice.services
```

```

class VideoResolution : Comparable<VideoResolution> {
    val width: Int
    val height: Int

    constructor(resolution: String) {

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.02.07-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

        val r = resolution.split("x".toRegex()).toTypedArray()
        width = r[0].toInt()
        height = r[1].toInt()
    }

    constructor(width: Int, height: Int) {
        this.width = width
        this.height = height
    }

    private fun getBandwidth(): Int {
        return when (height) {
            240 -> 246440
            360 -> 460560
            480 -> 836280
            720 -> 2149280
            1080 -> 6221600
            else -> 6221600
        }
    }

    fun getBaseFileText(): String {
        return "#EXT-X-STREAM-INF:" +
            "PROGRAM-ID=1," +
            "BANDWIDTH=${getBandwidth()}," +
            "RESOLUTION=${width}x$height," +
            "NAME=\"\$height\"\\n\" +
            "${height}p/video.m3u8"
    }

    override fun toString(): String {
        return width.toString() + "x" + height
    }

    override fun compareTo(other: VideoResolution): Int {
        return width - other.width
    }
}

```

2.5. VideoService.kt

```
package tv.comnata.videoservice.services
```

```

import feign.FeignException
import org.slf4j.LoggerFactory
import org.springframework.beans.factory.annotation.Autowired
import org.springframework.stereotype.Service
import org.springframework.web.multipart.MultipartFile

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.02.07-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

import tv.comnata.videoservice.clients .MainClient
import tv.comnata.videoservice.entities .VideoUploadResponse
import tv.comnata.videoservice.entities .VideoUploadResponseError
import tv.comnata.videoservice.entities .VideoUploadResponseSuccess
import tv.comnata.videoservice.services .FfmpegManager.OnUpdateProgressListener
import java.io .File
import java.io .IOException
import java.nio.charset .StandardCharsets
import java.nio .file .Files
import java.nio .file .Path
import java.nio .file .Paths
import java.util.*

@Service
class VideoService(
    @Autowired
    private var mainClient: MainClient
) : OnUpdateProgressListener {
    private fun createDirectoryIfNotExists(realPath: String) {
        val theDir = File(realPath)
        if (!theDir.exists()) {
            theDir.mkdirs()
        }
    }

    private fun createBaseFile(path: String, videoId: String, videoResolution: VideoResolution)
    {
        val resolutions = listOf(240, 360, 480, 720, 1080)

        val contentBuilder = arrayListOf<String>()
        contentBuilder.add("#EXTM3U")

        for (resolution in resolutions) {
            if (videoResolution.height >= resolution) {
                contentBuilder.add(
                    VideoResolution(
                        videoResolution.width / videoResolution.height * resolution,
                        resolution
                    ).getBaseFileText()
                )
            }
        }

        val file : Path = Paths.get("$path$videoId/video.m3u8")
        Files.write( file , contentBuilder, StandardCharsets.UTF_8)
    }

    private fun createWorkDirectories(realPath: String, videoId: String) {

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.02.07-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

createDirectoryIfNotExists(realPath)
createDirectoryIfNotExists(realPath + videoId)
createDirectoryIfNotExists("$realPath$videoId/240p")
createDirectoryIfNotExists("$realPath$videoId/360p")
createDirectoryIfNotExists("$realPath$videoId/480p")
createDirectoryIfNotExists("$realPath$videoId/720p")
createDirectoryIfNotExists("$realPath$videoId/1080p")
}

fun saveVideo( file : MultipartFile, realPath: String): VideoUploadResponse {
    val separatedName = file.originalFilename!!. split(".")
    val videoUuid = UUID.randomUUID().toString()
        .replace("-", "")
        .substring(0, 6)
        .toUpperCase()
    val type = "." + separatedName[separatedName.size - 1]

    return try {
        mainClient.createVideo(videoUuid)
        if (!file.isEmpty && separatedName.size > 1) {
            createWorkDirectories(realPath, videoUuid)
            file.transferTo( File("$realPath$videoUuid/original$type"))

            val ffmpegManager = FfmpegManager("$realPath$videoUuid/", "original$type",
                this)
            createBaseFile(realPath, videoUuid, ffmpegManager.videoResolution)

            ffmpegManager.start()

            return VideoUploadResponseSuccess(videoUuid, "/video/getVideo/$videoUuid/
                video.m3u8")
        }
        VideoUploadResponseError("File is empty.")
    } catch (exception: IOException) {
        VideoUploadResponseError(exception.message!!)
    } catch (exception: FeignException) {
        VideoUploadResponseError(exception.message!!)
    }
}

override fun onUpdatePercent(videoUuid: String, percent: Double) {
    logger.info("Video $videoUuid: ${"%0.2f".format(percent)}%")
    mainClient.setVideoProgress(videoUuid, (percent * 100).toInt())
}

companion object {
    private val logger = LoggerFactory.getLogger(VideoService::class.java)

```

ИЗМ.	Лист	№ докум.	Подп.	Дата
RU.17701729.02.07-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата


```

        const val DIRECTORY_PATH = "videos"
    }
}

```

2.6. VideoUploadResponse.kt

```
package tv.comnata.videoservice.entities
```

```

abstract class VideoUploadResponse(
    val status: VideoUploadStatus,
) {
    enum class VideoUploadStatus {
        SUCCESS,
        FAILED,
    }
}

```

```

data class VideoUploadResponseSuccess(
    val videoId: String,
    val videoUrl: String,
) : VideoUploadResponse(VideoUploadStatus.SUCCESS)

```

```

data class VideoUploadResponseError(
    val message: String,
) : VideoUploadResponse(VideoUploadStatus.FAILED)

```

2.7. MainClient.kt

```
package tv.comnata.videoservice.clients
```

```

import org.springframework.cloud.openfeign.FeignClient
import org.springframework.web.bind.annotation.RequestMapping
import org.springframework.web.bind.annotation.RequestMethod
import org.springframework.web.bind.annotation.RequestParam

```

```

@FeignClient("main-service")
interface MainClient {
    @RequestMapping(value = ["/main/createVideo"], method = [RequestMethod.PUT])
    fun createVideo(@RequestParam videoUuid: String?): String?

    @RequestMapping(value = ["/main/setVideoProgress"], method = [RequestMethod.POST])
    fun setVideoProgress(@RequestParam videoUuid: String?, @RequestParam videoProgress: Int)
        : String?
}

```

2.8. AppConfig.java

```
package tv.comnata.videoservice.configs;
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.02.07-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```
import org.springframework.boot.web.servlet.MultipartConfigFactory;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;
import org.springframework.util.unit.DataSize;

import javax.servlet.MultipartConfigElement;

@Configuration
@ComponentScan("tv.comnata.videoservice")
public class AppConfig {
    @Bean
    MultipartConfigElement multipartConfigElement() {
        MultipartConfigFactory factory = new MultipartConfigFactory();

        factory.setMaxFileSize(DataSize.ofGigabytes(7));
        factory.setMaxRequestSize(DataSize.ofGigabytes(7));

        return factory.createMultipartConfig();
    }
}
```

2.9. JWTTokenStoreConfig.java

```
package tv.comnata.videoservice.configs;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.Primary;
import org.springframework.security.oauth2.provider.token.DefaultTokenServices;
import org.springframework.security.oauth2.provider.token.TokenStore;
import org.springframework.security.oauth2.provider.token.store.JwtAccessTokenConverter;
import org.springframework.security.oauth2.provider.token.store.JwtTokenStore;

@Configuration
public class JWTTokenStoreConfig {
    @Bean
    public TokenStore tokenStore() {
        return new JwtTokenStore(jwtAccessTokenConverter());
    }

    @Bean
    @Primary
    public DefaultTokenServices tokenServices() {
        DefaultTokenServices defaultTokenServices = new DefaultTokenServices();
        defaultTokenServices.setTokenStore(tokenStore());
        defaultTokenServices.setSupportRefreshToken(true);
    }
}
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.02.07-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

        return defaultTokenServices;
    }

    @Bean
    public JwtAccessTokenConverter jwtAccessTokenConverter() {
        JwtAccessTokenConverter converter = new JwtAccessTokenConverter();
        converter.setSigningKey("helloworld");
        return converter;
    }
}

```

2.10. ResourceServerConfiguration.java

```

package tv.comnata.videoservice.configs;

import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.oauth2.client .OAuth2ClientContext;
import org.springframework.security.oauth2.client .OAuth2RestTemplate;
import org.springframework.security.oauth2.client .resource .OAuth2ProtectedResourceDetails;
import org.springframework.security.oauth2.config.annotation.web.configuration.
    ResourceServerConfigurerAdapter;

@Configuration
public class ResourceServerConfiguration extends ResourceServerConfigurerAdapter {
    @Bean
    public OAuth2RestTemplate oauth2RestTemplate(
        @Qualifier("oauth2ClientContext") OAuth2ClientContext oauth2ClientContext,
        OAuth2ProtectedResourceDetails details
    ) {
        return new OAuth2RestTemplate(details, oauth2ClientContext);
    }

    @Override
    public void configure(HttpSecurity http) throws Exception {
        http.authorizeRequests()

            .anyRequest().permitAll();
    }
}

```

2.11. SwaggerConfig.kt

```

package tv.comnata.videoservice.configs

import org.springframework.context.annotation.Bean

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.02.07-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

import org.springframework.context.annotation.Configuration
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer
import springfox.documentation.builders.PathSelectors
import springfox.documentation.builders.RequestHandlerSelectors
import springfox.documentation.service.*
import springfox.documentation.spi.DocumentationType
import springfox.documentation.spring.web.plugins.Docket
import java.util.*

@Configuration
class SwaggerConfig : WebMvcConfigurer {
    @Bean
    fun api(): Docket? {
        val contact = Contact(
            "Vladislav Annenkov",
            "https://t.me/Vakosta",
            "v.a.annenkov@ya.ru"
        )

        val vext: List<VendorExtension<*>> = ArrayList()
        val apiInfo = ApiInfo(
            "Comnata Video API",
            "API for main features of the Comnata.",
            "1.0.0",
            "https://something.com",
            contact,
            "MIT",
            "https://something.com",
            vext
        )

        return Docket(DocumentationType.SWAGGER_2)
            .apiInfo(apiInfo)
            .forCodeGeneration(true)
            .securitySchemes(listOf(apiKey()) as List<SecurityScheme>?)
            .select()
            .apis(RequestHandlerSelectors.basePackage("tv.comnata.videoservice.controllers"))
            .paths(PathSelectors.any())
            .build()
    }

    override fun addResourceHandlers(registry: org.springframework.web.servlet.config.annotation.ResourceHandlerRegistry) {
        registry.addResourceHandler("swagger-ui.html")
            .addResourceLocations("classpath:/META-INF/resources/")
        registry.addResourceHandler("/webjars/**")
            .addResourceLocations("classpath:/META-INF/resources/webjars/")
    }
}

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.02.07-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```
private fun apiKey(): ApiKey {  
    return ApiKey("authkey", "Authorization", "sdf")  
}  
}
```

2.12. WebSecurityConfig.java

```
package tv.comnata.videoservice.configs;  
  
import org.springframework.security.config.annotation.web.builders.HttpSecurity;  
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;  
import org.springframework.security.config.annotation.web.configuration.  
    WebSecurityConfigurerAdapter;  
  
@EnableWebSecurity  
public class WebSecurityConfig extends WebSecurityConfigurerAdapter {  
    @Override  
    protected void configure(HttpSecurity http) throws Exception {  
        http.cors();  
    }  
}
```

2.13. application.yml

```
spring:  
  application:  
    name: video-service  
  cloud:  
    kubernetes:  
      enabled: false  
  
server:  
  port: 8190  
  servlet:  
    context-path: /video  
  
eureka:  
  client:  
    service-url:  
      defaultZone: ${EUREKA_URI:http://localhost:8761/eureka}  
    enabled: true  
  
security:  
  oauth2:  
    resource:  
      userInfoUri: http://localhost:8880/auth/user  
    client:
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.02.07-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

client-id: flametoken
client-secret: thisissecret

3. Текст программы сервиса EurekaServer

3.1. application.yml

```
server.port=8761
eureka.client.registerWithEureka=false
eureka.client.fetchRegistry=true
eureka.server.enable-self-preservation=true
# spring.cloud.config.username=root
# spring.cloud.config.password=s3cr3t
# spring.cloud.config.name=conf
# spring.cloud.config.uri=http://localhost:8081
```

4. Текст программы сервиса GatewayServer

4.1. application.yml

```
spring:
  application:
    name: gateway-server
  cloud:
    gateway:
      routes:
        - id: websocket-server
          uri: http://localhost:8188
          predicates:
            - Path=/ws/**
        - id: main-service
          uri: http://localhost:8192
          predicates:
            - Path=/main/**, /ws/**
        - id: video-service
          uri: http://localhost:8190
          predicates:
            - Path=/video/**
      discovery:
        locator:
          lower-case-service-id: true
server:
  port: 8762
  # ssl:
  #   key-store: classpath:keystore.p12
  #   key-store-password: qwerty
  #   key-alias: test_key
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.02.07-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```
#    key-store-type: PKCS12
eureka:
  instance:
    preferIpAddress: true
  client :
    registerWithEureka: false
    fetchRegistry: true
    serviceUrl:
      defaultZone: ${EUREKA_URI:http://eureka-server:8761/eureka}
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.02.07-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

Лист регистрации изменений

[illegible]