«Национальный исследовательский университет
«Высшая школа экономики»

Лицей

Индивидуальная выпускная работа

Проект “Таск-менеджер LB Tool”

Выполнил Акопян Артём Хачатурович

Москва, 2024

1. Описание проекта

Мой проект является десктопным приложением, предназначенным для лицейского СПД-проекта Lyceum Bells, отвечающего за управление звонками в зданиях Лицея. Этот же проект является заказчиком моего продукта.

Перед началом выполнения проекта у меня был базовый опыт бэкэнд-разработки и минимальный опыт SRE. Подобных проектов в прошлом не было.

2. Проблемное поле

Проблемным полем является сложность в управлении столь большим проектом, используя групповые чаты в мессенджере Telegram – сейчас используется именно это решение. Доказательством этой проблемы является затрачиваемое на соответствующие бизнес-процессы количество человеко-часов – около 3–4 в неделю, как сообщил один из координаторов проекта. Мой продукт решает эту проблему, минимизируя социальную составляющую и нивелируя риски, связанные с человеческим фактором – всё необходимое для координаторов доступно на одном экране. По моим подсчётам, это снижает временные затраты до приблизительно 1 часа в неделю, что является ощутимой экономией ресурсов. Также в силу гибкости и масштабируемости моего продукта есть перспектива дальнейшего расширения функционала, а также предоставления доступа к этому решению другим СПД-проектам Лицея.

3. Техническое задание

От главы Lyceum Bells было получено техническое задание, подразумевающее следующие функциональные требования:

1. Панель для актуальных задач
2. Раздел «Звонки» для саунд-дизайнеров
    a. Загрузка аудиофайлов со звонками
    b. Просмотр всех загруженных звонков
    c. Скачивание файлов со звонками
3. Раздел «Дизайны» для дизайнеров
    a. Загрузка медиафайлов с дизайнами
    b. Просмотр всех загруженных дизайнов
    c. Скачивание файлов с дизайнами
    d. Предпросмотр дизайнов
4. Раздел «Тексты» для текстовиков
    a. Загрузка текстов
    b. Просмотр текстов
5. Раздел для администраторов, где есть возможность видеть все актуальные задачи, управлять ими, управлять тематиками звонков (тематика – определённый период, сопровождающийся тем или иным мотивом звонков в Лицее), пользователями.
6. Возможность выхода из учётной записи
7. Экран регистрации
8. Экран входа

4. Функциональные возможности

Мой проект представляет и решает следующие функциональные возможности и проблемы:

1. Загрузка аудио – упрощает проверку материалов и доступ к ним
2. Загрузка дизайнов (креативов) – упрощает проверку материалов и доступ к ним
3. Загрузка текстов – упрощает проверку материалов и доступ к ним
4. Создание задач* – позволяет автоматизировать контроль для координаторов и уменьшить количество пропущенных сроков за счёт централизации
5. Создание тематик (далее – «волн») * – создаёт перспективу создания функции фильтрации и сортировки по волнам
6. Создание приглашений для пользователей, используя e-mail* – позволяет ограничить доступ к ПО до определённого круга лиц – участников проекта, имеющих код приглашения
7. Аутентификация
    a. *Регистрация учётной записи*
    b. *Авторизация с помощью логина/пароля*
8. Просмотр всех звонков – позволяет упростить контроль и доступ
9. Просмотр всех креативов – позволяет упростить контроль и доступ
10. Просмотр всех текстов – позволяет упростить контроль и доступ
11. Просмотр всех задач всех исполнителей* – позволяет упростить контроль
12. Управление пользователями* – позволяет упростить контроль и повысить уровень безопасности
13. Установка/замена фотографии пользователя (далее – аватарки) – позволит добавить персонализацию
14. Выход из учётной записи

5. Аналоги

Сейчас в проекте используется один из аналогов – групповые Telegram-чаты по всем отделам. Этот инструмент имеет минусы, в числе которых сложность в отслеживании и управлении выполнения задач. Нет чёткой структуры, трекинга статусов и единой библиотеки материалов. Это является ярко выраженной проблемой.

Также существуют многочисленные скрам-доски (Jira, Asana, Miro), платформы для командной работы и подобные (JetBrains Space, GitHub, GitLab), однако они все предоставляют возможность лишь отслеживать статус задач, что не является достаточным функционалом, ибо требуется также возможность загрузки файлов, управления доступом и так далее.

В отличии от вышеупомянутых сервисов моё решение полностью соответствует нуждам проекта, а также легко масштабируется как архитектурно, так и функционально, что делает его идеальным вариантом.

6. Использованные технологии

В моём проекте есть две части: back-end и клиентская (front-end и desktop).

Являясь back-end-разработчиком, я начал работу именно с него. Оно было реализовано на следующем стеке технологий:

- **Java** – основной язык программирования.
- **Kotlin** – дополнительный язык программирования, исполняющийся на JVM (Java Virtual Machine – виртуальная машина, создающаяся JRE, – Java Runtime Environment – на которой исполняется байт-код, скомпилированный из кода Java/Kotlin/Scala и прочих языков), что обеспечивает его полную совместимость с Java.
- **JavaScript** – скриптовый язык программирования.
- **Node.JS** – платформа, которая позволяет запускать JavaScript-код вне браузера.
- **Spring Framework** – фреймворк с открытым исходным кодом для разработки приложений на языке Java.
- **Spring Boot** – дополнение к Spring Framework, созданное для упрощения и ускорения разработки приложений.
- **Spring Core** – основной модуль Spring Framework, который отвечает за управление зависимостями и IoC (Inversion of Control – принцип программирования, использующийся для уменьшения связанности кода.
- **Spring Web** – модуль Spring Framework, которая предоставляет инструменты и компоненты для создания веб-приложений.
- **Spring Security** – фреймворк, являющийся модулем Spring Framework, для аутентификации и авторизации.
- **Spring Data JPA** – модуль Spring Framework, являющаяся расширением технологии JPA (Java Persistence API), предназначенное для упрощения работы с реляционными базами данных.

- **Spring Mail** – модуль Spring Framework, который предоставляет инструменты для работы с электронной почтой.
- **Hibernate** – фреймворк для языка программирования Java, предназначенный для реализации объектно-реляционного отображения.
- **AWS SDK** – библиотека, предназначенная для взаимодействия с AWS-based API.
- **OpenAPI** – спецификация, которая определяет стандартный, языково-независимый интерфейс для описания RESTful API.
- **Swagger UI** – интерактивный веб-интерфейс для визуализации и тестирования RESTful API.
- **JWT** (JSON Web Token) – стандартный формат токенов, который используется для аутентификации и авторизации в веб-приложениях.
- **Micrometer** – библиотека, предназначенная для создания и управления метриками в приложениях.
- **Prometheus** – система мониторинга и база данных временных рядов для сбора и хранения метрик.
- **Grafana** – платформа для визуализации и мониторинга данных с возможностью создания дэшбордов.
- **Yandex Object Storage** – высокоэффективный облачный сервис для хранения объектов, вдохновлённый AWS S3.
- **Express.js –** минималистичный и гибкий веб-фреймворк для Node.js, предназначенный для создания несложных веб-приложений.
- **Axios** – библиотека для выполнения HTTP-запросов в JavaScript.
- **PostgreSQL** – объектно-реляционная система управления базами данных с открытым исходным кодом, которая использует и расширяет язык SQL
- **Docker** – платформа для разработки, доставки и запуска контейнерных приложений.

- **Docker Compose** – инструмент, который упрощает создание и управление многоконтейнерными приложениями в Docker.
- **GitHub Actions** – платформа автоматизации, встроенная в GitHub, которая предоставляет средства для реализации непрерывной интеграции и непрерывной доставки (CI/CD).

В свою очередь клиентская часть реализована с использованием следующих технологий:

- **Kotlin** – основной язык программирования.
- **Kotlin Multiplatform** – технология, разработанная компанией JetBrains, которая позволяет создавать кроссплатформенные приложения, используя общий код для различных платформ.
- **Compose Multiplatform** – фреймворк, разработанный компанией JetBrains, который позволяет создавать пользовательские интерфейсы для приложений, написанный на Kotlin Multiplatform.
- **Ktor** – фреймворк для создания серверных и клиентских приложений на языке Kotlin.
- **Ktor Client** – компонент фреймворка Ktor который предоставляет инструменты для создания HTTP-клиентов.
- **Ktor Serialization** – модуль в рамках фреймворка Ktor, который обеспечивает средства для сериализации и десериализации данных.
- **FileKit** – библиотека для Kotlin Multiplatform, предназначенная для упрощения работы с файлами и медиафайлами на различных платформах.
- **Nginx** – высокопроизводительный веб-сервер и обратный прокси-сервер.

7. Рефлексия

1. Проблемы

В ходе работы над проектом основной проблемой являлось время и планирование. В силу его некоторой масштабности пришлось ощутимо изменить свой образ жизни, чтобы завершить выполнение к назначенному дню.

Также возникла проблема при проектировании и написании клиентской части – ранее был минимальный опыт в этой области, а Compose, который был основным в клиентском стеке, является декларативным фреймворком, в котором у меня до сего проекта абсолютно не было экспертизы, в связи с чем пришлось изучать выбранную технологию почти с нуля, ровно как UI-/UX-дизайн. В изучении моего набора мне помогла документация – все использованные мной библиотеки и фреймворки были исчерпывающе задокументированы.

Несколько проблем также всплывали в процессе сборки проекта, в связи с чем приходилось менять некоторые модули или использовать концепцию декларации except-actual, которая позволяет указать определённое поведение для конкретного модуля.

В серверной же части проблем, как ни странно, не возникало, так как Spring достаточно задокументирован, а также в интернете немало информации по модулям или возникающим ошибкам.

2. Перспективы

В качестве перспектив я вижу следующие обновления:

- Добавление фильтров в каждый из разделов.
- Добавление интеграции с Электронным Журналом/платформой 2359.
- Создание мобильной версии (приложения).
- Создание лэндинга с документацией по использованию.
- Перемещение серверной части на более мощную виртуальную машину для увеличения производительности.

- Выдача другим СПД-проектам Лицея этого продукта для свободного использования.
- Упрощение регистрационной формы с использованием Единого личного кабинета НИУ ВШЭ.
- Интеграция с API стриминговых сервисов/иных каталогов музыкальных треков для упрощённого ввода названия/исполнителя трека.
- Создание экрана для техников, на котором будет возможность сформировать архив из треков в соответствии с форматом.
- Расширить платформу, добавив и интегрировав общий портал выбора звонков для всех лицеистов, где будет возможность предлагать треки и голосовать за понравившиеся варианты. Полная интеграция позволит ещё больше минимизировать временные затраты и оптимизировать бизнес-процессы до стадии полной автономии от сторонних сервисов.

3. Приобретенные навыки

За время работы над проектом я обрёл и улучшил многие навыки. Буду использовать общепринятое деление на soft (личностные качества) и hard (профессиональные знания).

Soft skills:

- Планирование
- Стрессоустойчивость
- Поиск информации в интернете
- Критическое мышление
- Креативность
- Адаптивность
- Коммуникабельность

Hard skills:

- UI/UX
- Архитектурное проектирование
- Docker и Docker Compose
- GitHub Actions
- Kotlin
- Java Persistence API
- Spring Security
- JWT
- Grafana
- Prometheus
- AWS S3
- Hibernate
- Spring Mail
- Compose
- Ktor
- Nginx
- DI

Указанные навыки сильно мне помогут в дальнейшей учёбе и карьере самым прямым образом: приобретённые профессиональные навыки сейчас являются неотъемлемой частью индустрии информационных технологий, то есть актуальны и востребованы. Личностные же качества способны обеспечить комфортную и эффективную работу в команде и успех в построении моего будущего, в том числе при подготовке к ЕГЭ и к сессиям в вузе.

4. Ретроспектива

Оглядываясь назад, при написании клиентской части я бы отказался от Kotlin Multiplatform в пользу фреймворка .NET на C#, ибо по опыту людей, с которыми мне довелось взаимодействовать в ходе работы над проектом, и тому, что я получил в процессе разработки, модули для десктопной и веб-разработки Kotlin Multiplatform не являются production ready решением и не подходят для полноценной реализации продукта «с нуля».

Также, если бы я обладал кратно большим количеством времени, я бы реализовал серверную часть с использованием микросервисов на различных технологиях, так как это повысило бы масштабируемость и гибкость инфраструктуры.

Хотелось бы отметить, что я жалею о том, что начал работу над проектом так поздно. Это вызвало множество проблем и трудностей, не все из которых было легко преодолеть. В ретроспективе я бы начал делать этот проект около июля-августа, так как мне кажется это оптимальным количеством времени для построения такого сервиса без проблем с тайм-менеджментом.

8. Приложения

GitHub – Back-end: https://github.com/isntrui/lb (в корни репозитория находится ссылка на клиент)

GitHub – Client: https://github.com/isntrui/lb_client