Java Project 0

TEL-RAN
by Starta Institute

Финальные проекты на Java

Цели, архитектура, стек, плюсы и минусы, выводы и минимальные требования

Идеи для имплементации

- Банковское приложение
- Java чат-бот Телеграмм
- Сервис для сокращения URL
- Приложение для заказа пиццы

Проект - Банковский микросервис

TEL-RAN
by Starta Institute

Цели проекта "Банковский микросервис":

1. Централизованное управление: Создание микросервиса, который позволит банковским учреждениям управлять операциями своих клиентов, такими как переводы, проверка баланса и управление профилями клиентов.
2. Оптимизация банковских операций: Автоматизация основных банковских операций, что ускорит процесс обработки транзакций и улучшит обслуживание клиентов.
3. Безопасность и конфиденциальность: Обеспечение высокого уровня безопасности для всех банковских операций и данных клиентов, используя современные методы шифрования и аутентификации.
4. Масштабируемость: Разработка микросервиса таким образом, чтобы он мог обслуживать большое количество клиентов одновременно, обеспечивая стабильную работу даже при высоких нагрузках.
5. Интеграция с другими системами: Возможность легкой интеграции с другими банковскими и финансовыми системами, такими как системы обработки платежей или кредитные бюро.
6. Автоматизированное тестирование: Внедрение автоматических тестов для проверки корректности всех банковских операций и обеспечения высокого качества программного обеспечения.
7. Документирование API: Создание подробной документации для API микросервиса, что облегчит его интеграцию с другими системами и упростит работу разработчиков.
8. Отчетность и аналитика: Внедрение инструментов для сбора статистики и аналитики по банковским операциям, что позволит банкам лучше понимать потребности своих клиентов и оптимизировать свои услуги.
9. Обратная связь с клиентами: Возможность для клиентов оставлять отзывы или предложения по улучшению банковских услуг, что поможет банку улучшить качество своего обслуживания.

Проект - Java Telegram Bot

TEL-RAN
by Starta Institute

Цели проекта "Java Telegram Bot":

1. Многозадачность и интеграция: Разработка бота, который может интегрироваться с различными сторонними сервисами, такими как метеорологические службы, биржевые платформы и новостные агентства, чтобы предоставлять актуальную информацию подписчикам.
2. Персонализация контента: Возможность для пользователей настроить свои предпочтения, чтобы получать информацию, которая их интересует, например, погода в конкретном городе или новости по определенной теме.
3. Безопасность и конфиденциальность: Обеспечение безопасности данных пользователей и их настроек, а также обеспечение конфиденциальности переписки между ботом и пользователем.
4. Статистика для администратора: Внедрение функционала, который позволит администратору бота просматривать статистику по активности пользователей, популярности определенных запросов и другие ключевые метрики.
5. Масштабируемость: Построение архитектуры бота таким образом, чтобы он мог обслуживать большое количество пользователей одновременно, обеспечивая стабильную работу даже при высоких нагрузках.
6. Автоматизированные уведомления: Возможность для пользователей настроить автоматические уведомления о новой информации, основываясь на их предпочтениях.
7. Обратная связь: Внедрение функционала, который позволит пользователям оставлять отзывы или предложения по улучшению работы бота, что поможет улучшить качество предоставляемой информации и функционала.
8. Соблюдение стандартов Telegram: Обеспечение соответствия всех функций и возможностей бота стандартам и требованиям платформы Telegram.
9. Непрерывное обновление: Разработка механизма, который позволит быстро и легко обновлять информацию и добавлять новые источники данных, чтобы бот всегда предоставлял актуальную и релевантную информацию.

Проект - "Pizza"

Цели проекта "Pizza":

1. Разработка интегрированной системы управления: Создание системы, которая позволит администраторам управлять меню пиццерии, а также информацией о кафе, в которых эти пиццы доступны.
2. Оптимизация процессов: Основная идея заключается в том, чтобы автоматизировать процессы создания, чтения, обновления и удаления (CRUD) записей о пиццах и кафе. Это может значительно ускорить рабочие процессы.
3. Безопасность данных: Обеспечение безопасности данных клиентов и информации о продуктах с помощью современных методов аутентификации и авторизации.
4. Расширяемость: Создание системы таким образом, чтобы в будущем можно было легко добавлять новые функции или интегрироваться с другими системами.
5. Масштабируемость: Построение архитектуры таким образом, чтобы приложение могло обслуживать большое количество пользователей одновременно без сбоев.
6. Аналитика и отчетность: Внедрение инструментов для сбора статистики и аналитики, которые помогут администраторам понимать популярность различных пицц, частоту заказов и другие ключевые метрики.
7. Обратная связь: Возможность для клиентов оставлять отзывы о пицце или кафе, что поможет улучшить качество услуг и предложений.
8. Интеграция с внешними системами: (Опционально) Возможность интеграции с другими системами или платформами, такими как системы доставки или платежные системы.
9. Интерактивный интерфейс: (Опционально) Разработка простого и интуитивно понятного интерфейса для клиентов, чтобы они могли легко выбирать пиццу и кафе.
10. Экологичность: (Опционально) Внедрение функций, которые позволят клиентам выбирать экологически чистые варианты пиццы или кафе.

Проект - URL Shortener

TEL-RAN
by Starta Institute

Цели проекта "URL Shortener":

1. Эффективное сокращение URL: Разработка сервиса, который быстро и надежно преобразует длинные URL-адреса в короткие, удобные для использования ссылки.
2. Удобство использования: Создание интуитивно понятного интерфейса, который позволит пользователям без труда сокращать URL-адреса и управлять ими.
3. Безопасность: Обеспечение безопасности сокращенных URL-адресов, предотвращение возможности создания вредоносных или мошеннических ссылок через сервис.
4. Статистика и аналитика: Внедрение инструментов для отслеживания статистики переходов по сокращенным ссылкам, что позволит пользователям понимать популярность и эффективность их ссылок.
5. Производительность и масштабируемость: Разработка сервиса таким образом, чтобы он мог обрабатывать большое количество запросов одновременно и быстро возвращать результаты.
6. Постоянство и надежность: Обеспечение того, чтобы сокращенные URL-адреса были постоянными и всегда вели к нужному ресурсу, даже после длительного времени.
7. Интеграция с другими платформами: Возможность легкой интеграции сервиса с другими веб-сайтами и приложениями, чтобы пользователи могли сокращать URL-адреса прямо из интерфейсов этих платформ.
8. Расширяемость: Построение архитектуры сервиса таким образом, чтобы в будущем можно было легко добавлять новые функции и возможности.
9. Экологичность: Сокращение длинных URL-адресов помогает уменьшить объем передаваемых данных, что положительно сказывается на экологии, сокращая электроэнергию на передачу и обработку данных.

Анализ выбора

TEL-RAN
by Starta Institute

1. URL Shortener

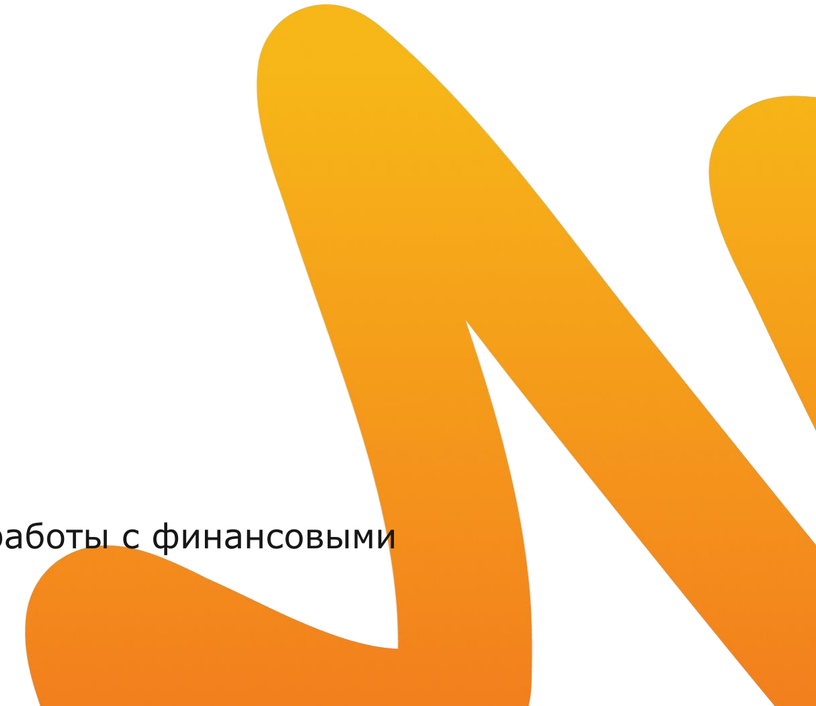
Плюсы:
Отличный проект для изучения основных принципов веб-разработки.
Реализация URL Shortener — популярная задача, которая может быть использована в реальной жизни.
Возможность изучить различные базы данных (MySQL, Mongo).

Минусы:
Относительно простой проект.

2. Backend Development of a Banking Microservice

Плюсы:
Комплексный проект, включающий в себя множество аспектов разработки.
Возможность изучить дополнительные технологии и инструменты.
Работа с финансовыми данными требует высокой степени безопасности и надежности.

Минусы:
Проект может потребовать более детального планирования и анализа из-за специфики работы с финансовыми данными.



Анализ выбора

TEL-RAN
by Starta Institute

3. Java Telegram Bot

Плюсы:
Работа с популярным мессенджером Telegram.
Возможность интеграции с различными сторонними сервисами.
Практическая польза — создание реального бота для пользователей.

Минусы:
Требует понимания работы API Telegram и сторонних сервисов.

4. Pizza Project

Плюсы:
Работа с реальным бизнес-кейсом — управление пиццерией.
Возможность изучить CRUD-операции в деталях.
Понимание взаимодействия между различными сущностями (пицца, кафе).
Ясный и понятный веб-интерфейс

Минусы:
Относительно простой проект без сложных интеграций или алгоритмов.

Выводы

TEL-RAN
by Starta Institute

Общий вывод:

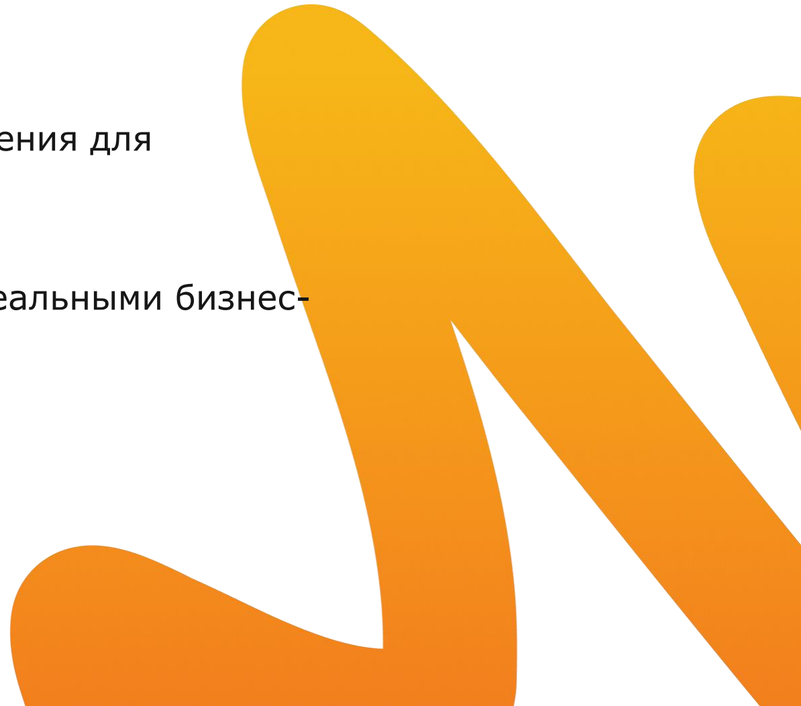
Все проекты предоставляют отличную возможность для закрепления изученного и освоения нового материала по Java и связанных технологий.
Выбор конкретного проекта зависит от ваших интересов.

Основы веб-разработки и архитектуры, URL Shortener — отличный выбор.

Множество технологий, Banking Microservice будет наилучшим вариантом.

Telegram Bot подойдет для всех: работа с популярными API и создание реального приложения для пользователей.

Pizza Project идеально подходит для изучения базовых операций разработки и работы с реальными бизнес-кейсами.



Свой проект

Свой проект должен соответствовать минимальному объёму, который показывает знания и навыки студента-выпускника (см. слайд Минимальные требования).

Типовые проекты – ориентиры, на которые нужно равняться при выборе темы своего проекта.

TEL-RAN
by Starta Institute

С чего начать описание идеи?

TEL-RAN
by Starta Institute

Декомпозируем задачу описания

Описать приложение целиком сразу очень сложно. Задачу описания нужно разделить (декомпозировать) на части.

1. Гексогональная архитектура предполагает, что ядром системы является доменная модель, т.е. область знания, для которой создаётся приложение (энергетика, ретейл, услуги и т.д.). Поэтому на первом этапе формулируем назначение приложения (сверхзадача) – то, какие проблемы оно должно решить.
2. В соответствии с назначением выбираем ёмкое имя проекта.
3. Понимая, что мы создаём, следует взглянуть на приложение глазами пользователя и составить перечень пожеланий пользователя. Предлагается составить набор *User Story (US)*. Когда US будут описаны, станет понятен объём реализуемого функционала.

User Story - это подход в гибкой методологии, позволяющий коротко сформулировать одно требование пользователя к приложению.
https://www.atlassian.com/ru/agile/project-management/user-stories
Примеры user story см. в Приложении А.

4. Составляем техническое задание, первым разделом которого будет описание доменной модели:

- бизнес-сущности (какие entity нужны, какие у них будут поля, какие связи друг с другом)
- бизнес-логику (действия, бизнес-правила, сценарии, исключительные ситуации). Эти описания лягут в логику классов, которыми будут пользоваться сервисы будущего приложения.

С чего начать описание идеи?

TEL-RAN
by Starta Institute

Декомпозируем задачу описания

5. После того, как сущности и их взаимоотношения определены следует подумать о том, какие данные нужно хранить длительно (слой *persistence*).
6. Определите, какие данные для ваших сущностей нужно получать от пользовательского приложения и какие данные нужно отдавать пользовательскому приложению. На основании этих требований определитесь с перечнем DTO и их полями, а также составом и методами Ваших контроллеров. На основании данного раздела позже нужно будет составить описание API в readme.md. Можно использовать swagger для автоматизированного описания API.
7. Подумайте о маппинге DTO в сущности и обратно.
8. Подумайте об исключительных ситуациях в приложении, способах их обработки и перехвата, а также соответствии http-кодам статусов и содержанию ответов сервера.
9. Создайте правила безопасности для Вашего приложения. Опишите, как правила будут реализовываться.
10. Подумайте о логировании, мониторинге и развёртывании Вашего приложения (какая инфраструктура нужна для его работы).
11. Проанализировав изложенное ранее, назначьте каждому разделу технологии, которые потребуются для выполнения поставленных задач.

Архитектура

TEL-RAN
by Starta Institute

1. Слой представления (**Presentation Layer**): Этот слой отвечает за интерфейс пользователя, будь то веб-интерфейс, мобильное приложение или API для сторонних систем. Он обрабатывает пользовательский ввод и отображает данные пользователю. Классы: DTO, контроллеры

2. Слой бизнес-логики (**Business Logic Layer**): Здесь содержится основная бизнес-логика приложения. Этот слой обрабатывает данные, полученные от слоя представления, выполняет необходимые вычисления или обработку и передает данные на следующий слой или возвращает их обратно в слой представления. Классы основной логики, Entity-классы.
3. Слой сервисов (**Service Layer**): Может быть добавлен между слоем представления и слоем бизнес-логики, особенно если есть необходимость в реализации бизнес-логики, которая должна быть доступна через несколько различных интерфейсов. Сервисы и мапперы для превращения сущностей в DTO.
4. Слой доступа к данным (**Data Access Layer**): Этот слой отвечает за взаимодействие с базой данных или другими внешними системами хранения данных. Он содержит все операции CRUD (создание, чтение, обновление, удаление) и обеспечивает абстракцию от конкретной системы хранения данных. Классы: Repository, DAO

Дополнительные компоненты:

5. Слой интеграции (**Integration Layer**): Если приложение интегрируется с внешними системами или сервисами, этот слой может обеспечивать необходимую абстракцию и трансформацию данных.

Универсальный стек технологий

TEL-RAN
by Starta Institute

- Язык программирования: Java (версии 17 или 21)
- Основной фреймворк: Spring Boot (последняя стабильная версия 3.x.x – без приписок SNAPSHOT или M1)
- Базы данных: MySQL или PostgreSQL или Mongo или H2 in-memory database. Допустимо развёртывание БД и другой инфраструктуры приложения в контейнере Docker.
- Миграция базы данных: Liquibase
- Тестирование: Mockito, JUnit 5
- Документация API: Swagger, Postman
- Документация кода: JavaDoc
- Maven (для сборки проекта)
- JaCoCo (для измерения покрытия кода тестами)
- Java, MapStruct или Kotlin (для маппинга объектов)
- Hibernate и JPA (для работы с базой данных)
- SLF4J (для логирования)
- GitHub (для хранения кода и документации)
- Word Office (для документации)
- Docker (опционально для развертывания)
- RestTemplate/RestClient, Kafka/Rabbit MQ (опционально)

Минимальные требования

TEL-RAN
by Starta Institute

- **Трехуровневая архитектура:** Каждый проект должен иметь разделение на уровни контроллера, сервиса и репозитория.
- **База данных:** Необходимо иметь базу данных для хранения и извлечения данных. Может быть реализована как реляционная (например, MySQL), так и NoSQL (например, MongoDB) база данных.
- **RESTful API:** Проекты должны предоставлять API для взаимодействия с функциональностью. Это включает в себя создание, чтение, обновление и удаление данных.
- **Безопасность:** Доступ к определенным функциям (например, добавление или удаление данных) должен быть ограничен и требовать аутентификации.
- **Тестирование:** Основные функции и методы должны быть покрыты юнит-тестами.
- **Документация:** Необходимо иметь документацию для API, описывающую доступные конечные точки, параметры и ожидаемые ответы.
- **Интеграция с внешними сервисами:** В зависимости от проекта, может потребоваться интеграция с внешними API или сервисами (например, Telegram API или сторонние сервисы для получения данных).
- **Миграция базы данных:** Для управления изменениями в структуре базы данных рекомендуется использовать инструменты миграции, такие как Liquibase или Flyway.
- **Логирование:** Все важные события и ошибки в приложении должны быть залогированы.

Назад

Приложение А. Примеры User Story

TEL-RAN
by Starta Institute

- Я как новый пользователь, хочу зарегистрироваться в приложении, чтобы начать его использовать.
- Я как пользователь, хочу войти в систему с использованием электронной почты и пароля, чтобы получить доступ к своему аккаунту.
- Я как пользователь, хочу иметь возможность восстановить забытый пароль через электронную почту, чтобы снова получить доступ к аккаунту.
- Я как пользователь, хочу просматривать свой профиль, чтобы видеть или редактировать свои личные данные.
- Я как покупатель в интернет-магазине, хочу добавлять товары в корзину, чтобы сформировать заказ.
- Я как покупатель, хочу выбирать способ оплаты при оформлении заказа, чтобы оплатить покупку удобным для меня способом.
- Я как пользователь социальной сети, хочу иметь возможность публиковать сообщения, чтобы делиться своими мыслями и идеями с друзьями.
- Я как пользователь мобильного приложения, хочу получать уведомления о важных событиях, чтобы быть в курсе происходящего.
- Я как администратор системы, хочу иметь возможность блокировать пользователей за нарушение правил, чтобы поддерживать порядок в системе.
- Я как пользователь, хочу иметь возможность изменить язык интерфейса, чтобы использовать программное обеспечение на предпочитаемом мной языке.
- Я как пользователь, хочу иметь возможность загрузить своё фото для профиля, чтобы персонализировать свой аккаунт.
- Я как пользователь, хочу иметь возможность искать других пользователей по имени или электронной почте, чтобы добавлять их в друзья или следить за их активностью.
- Я как клиент банка, хочу видеть историю своих транзакций в личном кабинете, чтобы контролировать свои финансы.
- Я как пользователь, хочу иметь возможность отключить уведомления от приложения, чтобы они не отвлекали меня в неподходящее время.
- Я как владелец бизнеса, хочу иметь возможность анализировать статистику посещаемости моего сайта, чтобы понимать поведение пользователей и улучшать сервис.

Назад