Разделы описания проекта

1. Назначение приложения.
   1.1. Общая вводная информацию (о чём речь)
   1.2. Перечень пользователей (user, manager, admin и т.д.)
   1.3. Какая польза от приложения будущим пользователям и владельцу приложения?
2. Основной функционал (по пунктам) на основании User Story, при ведённых в приложении
3. Доменная модель:
   3.1. Сущности (основные понятия и расшифровки, состав данных). Например,
      3.1.1.Пользователь (сразу закладываем поля password, username и role/roles для безопасности)
      3.1.2.подписка
      3.1.3.заказ (например, на составление URL) и т.д.
   3.2. бизнес-правила
      3.2.1.ограничение доступа при окончании подписки
      3.2.2.логика работы ядра приложения
      3.2.3.акции, бонусы и т.д.
   3.3. сервисы (как классы, реализующие работу с сущностями и бизнес-логику) (проверяем, что все функциональные требования имеют отражения в сервисах)
   3.4. опционально: сервисы, работающие по расписанию.
4. Хранение данных:
   4.1. Вид базы данных (реляционная, NoSQL)
   4.2. Описание репозиториев или DAO-классов
5. Обмен данными с пользователем
   5.1. REST API:
      5.1.1.DTO (назначение и состав)
      5.1.2.Контроллеры и их эндпоинты
   5.2. MVC (делаем после REST API, если остаётся время. Если времени не осталось, то пишем, что фронтенд будет реализован в будущем)
      5.2.1.Контроллеры представления
      5.2.2.Формы
      5.2.3.Шаблоны представлений
6. Безопасность
   6.1. Способ аутентификации
   6.2. Роли и эндпоинты, доступные этим ролям.
   6.3. Опционально: управление сессией.
7. Интеграции со сторонними сервисами (например, с платёжной системой, хранилищем файлов и т.д.)
   7.1. Назначение интеграции
   7.2. Краткое описание API (ссылки на документацию и какие эндпоинты применялись)
   7.3. Регистрация и аутентификация Вашего приложения в стороннем сервисе.
8. Поверка качества кода
   8.1. Unit-тесты (сервисы, классы доменной модели, мапперы)
   8.2. интеграционные тесты (репозитории, контроллеры)
   8.3. code-review
9. Развёртывание
   9.1. Что нужно, чтобы развернуть приложение на сервере (другом компьютере)
   9.2. Опционально: миграция данных (наполнение БД перед развёртыванием приложения)
10. Документирование
   10.1. Документирование кода

10.2. Документирование API

11. Будущее развитие проекта
12. Стек технологий в соответствии с каждым пунктом Выше. Например,

Функционал	**Применяемые технологии**
Основной фреймворк для организации приложения	Spring Framework 6 с модулем быстрого конфигурирования Spring Boot 3
Хранение данных	
СУБД	PostgreSQL
Доступ к данным	Репозитории Spring Data JPA на базе Hibernate
Обмен данными с пользователем	
Организация контроллеров	Spring Web MVC
Организация представлений	Thymeleaf
Маппинг объектов в JSON	Jackson
Безопасность	
Организация аутентификации и доступа по ролям	Spring Security
Интеграции со сторонними сервисами	
Организация REST-клиента	Spring Cloud OpenFeign
Поверка качества кода	
Тестирование кода	JUnit5, Mockito
Оценка покрытия кода тестами	Средства IntelliJ IDEA
Code-review	Средства GitHub
Развёртывание	
Развёртывание СУБД (если предполагается просто установка, можно не писать)	Docker
Наполнение БД (миграция)	Liquibase
Документирование	
Документирование кода	JavaDoc
Документирование API	OpenAPI v3 (Swagger), SpringDoc

Памятка по прохождению code-review

1. Добавляете меня в репозиторий GitHub как участника команды https://github.com/KostjanoyYA
2. Делаете коммит и пуш текущего состояния проекта.
3. Создаёте себе список задач (хотя бы на листочке)
4. Создаёте вторую git-ветку от текущего состояния
5. Выполняете в рамках этой ветки одну задачу из списка (одна ветка на одну задачу). Там же делаете тесты на выполненный функционал.
6. После коммитов и пуша в новую ветку выставляете pull request для влития изменений из новой ветки в основную (main или master). Меня указываете в поле reviwer.
7. Я провожу code-review.
8. Вы смотрите замечания, вносите изменения в ветке, где выполняли доработку, делаете коммиты и затем пуш. Pull request обновится.
9. Я смотрю повторно и подтверждаю pull request.
10. Вы вливаете изменения в основную ветку в интерфейсе GitHub.