Архитектурный документ

Таксопарк «Бешеная черепаха»

Команда 5

# Раздел регистрации изменений

|  |  |  |  |
| --- | --- | --- | --- |
| **Версия документа** | **Дата** | **Описание изменения** | **Автор** |
| 1.0 | 27.05.21 | Первая версия | Пичугин А.Н. |
|  |  |  |  |

# Введение

## Название проекта

Таксопарк «Бешеная черепаха»

## Рамки проекта (Scope)

От внедряемой системы требуется предоставление информации по доходам от услуг и по закрузке машин – как собственных, так и внешних водителей. При некачественном обслуживании – не верно указаны данные по заказу и например, машина пришла не туда, - нужно знать имя диспетчера, принявшего заказ. Нужен учет опозданий машин – кто, когда и почему, - и требуется статистика по случаям передачи заказа в другой таксопарк.

Кроме этого следует решить проблему быстрого автоматического оповещения клиентов о назначенных на его заказ машинах и о том, что машина уже на месте - без участия диспетчера. Для этого будет использоваться смс-сообщение на мобильный телефон клиента (если указан мобильный телефон) и требуется формирование автоматического голосового сообщения при автонаборе телефонного номера клиента – при информировании, что машина уже подана. Для отправки смс следует использовать интерфейс, предоставленной компанией «Лайф-Телекон», с которой заключен соответсвующий договор. Таксист может использовать свой мобильный телефон (с номером «Лайф-Телекон») для доступа к данным по «свободным» заказам, и он может с телефона указать «свое согласие» взять конкретный заказ.

Диспетчер может регистрировать и изменять заказы в системе.

Водитель может выбирать, какой заказ он хотел бы выполнить, а также изменять информацию о текущем выполняемом заказе.

# Общее описание архитектуры, задействованные архитектурные представления

Структура описания архитектуры в этом документе:

* Архитектурные факторы - **стр. 5**
* Технические описания архитектурных решений - **стр. 6-7**
* Представления архитектуры - **стр. 8-24**
  + Представление прецедентов - стр. 8
  + Логическое представление - стр. 9
  + Представление архитектуры процессов - стр. 10-21
  + Физическое представление - стр. 21
  + Представление развёртывания - стр. 21
  + Представление архитектуры данных - стр. 22-23
  + Представление реализации - стр. 23
  + Представление разработки - стр. 24
  + Нефункциональные аспекты - стр. 24

# Архитектурные факторы (цели и ограничения)

Руководство получает детальное представление об эффективности работы таксопарка за счёт внедрения централизованного учёта движения денежных средств, автоматизации вычисления экономических метрик.

Работники тратят меньше времени на организационные процессы за счёт автоматизации оформления заказов, сообщения об изменениях клиенту и пр.

Клиенты получают ускоренную обработку заказов, чувствую большую удовлетворённость сервисом, что заставляет их чаще возвращаться к нашей компании, а не уходить к конкурентам.

Цели:

1. Автоматизация организационных моментов
2. Автоматический сбор и обработка данных о происходящих денежных транзакциях

Ограничения:

1. Стабильность работы
2. Надёжность
3. Безопасность

# Технические описания архитектурных решений

## Техническое описание №1

### Проблема

Как должно быть обеспечено хранение информации в системе?

### Идея решения

Следует использовать стандартные средства JDBC с драйвером PostgreSQL

### Факторы

Требования ТЗ позволяют использовать стандартные средства, так как не требуется особенной производительности

Информация не будет утеряна при корректной реализации

### Решение

Использовать СУБД с открытым исходным кодом PostgreSQL. Для реализации взаимодействия сереверного кода с базой данных применяется стандартный механизм взаимодейтсвия Java программ с базами данных - JDBC - с использованием соответсвующего драйвера (PostgreSQL).

### Мотивировка

Работает с Java, на которой пишется проект. Общедоступное решение, хорошо знакомое команде разработки, что позволит быстро реализовать эту часть системы

### Неразрешенные вопросы

Нет

### Альтернативы

Вариант с NoSQL был отвергнут из-за малого опыта работы команды разработки с эти типом хранения данных

## Техническое описание №2

### Проблема

Как построен графический интерфейс системы для диспетчеров?

### Идея решения

Следует использовать графический фреймворк JavaFX

### Факторы

Требования ТЗ позволяют использовать стандартные средства

Широкие возможности для модифицирования внешнего вида

Удобство работы

### Решение

Следует использовать графический фреймворк JavaFX, предоставляющий широкий набор готовых библиотек. Благодаря практикам реактивного программирования, заложенным в библиотеку, позволит разработать динамичный графический интерфейс.

### Мотивировка

Работает с Java и Kotlin, на которой пишется проект, общедоступное решение, хорошо знакомое команде разработки, что позволит быстро реализовать эту часть системы

### Неразрешенные вопросы

Нет

### Альтернативы

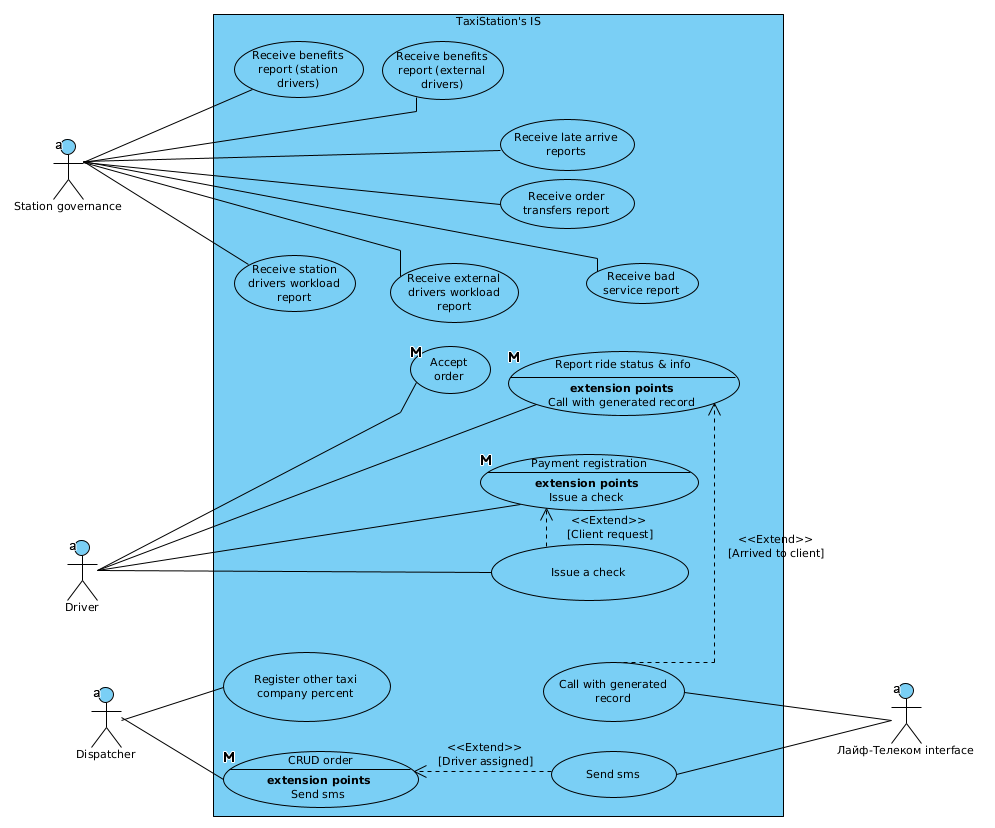
Вариант с Jetpack Compose был отвергнут из-за нестабильности текущей версии этого фреймворка, малом количестве доступных функций.

Swing отвергнут из-за меньшего опыта работы команды разработки с этим фреймворком

# Представления архитектуры

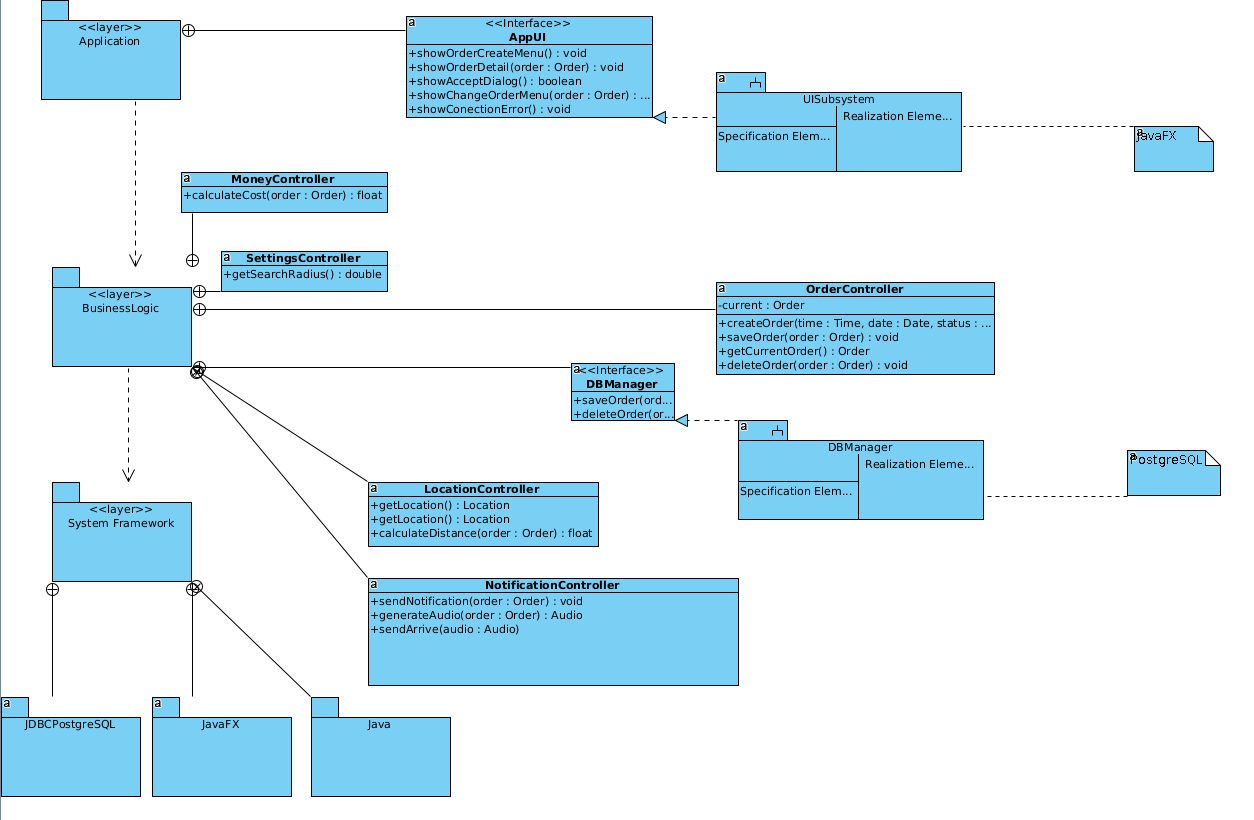
## Представление прецедентов (сценариев использования)

Диаграмма прецедентов:



## Логическое представление архитектуры

Слои и подсистемы:



## Представление архитектуры процессов

Диаграмма последовательности CRUD заказа (основной поток, Create):

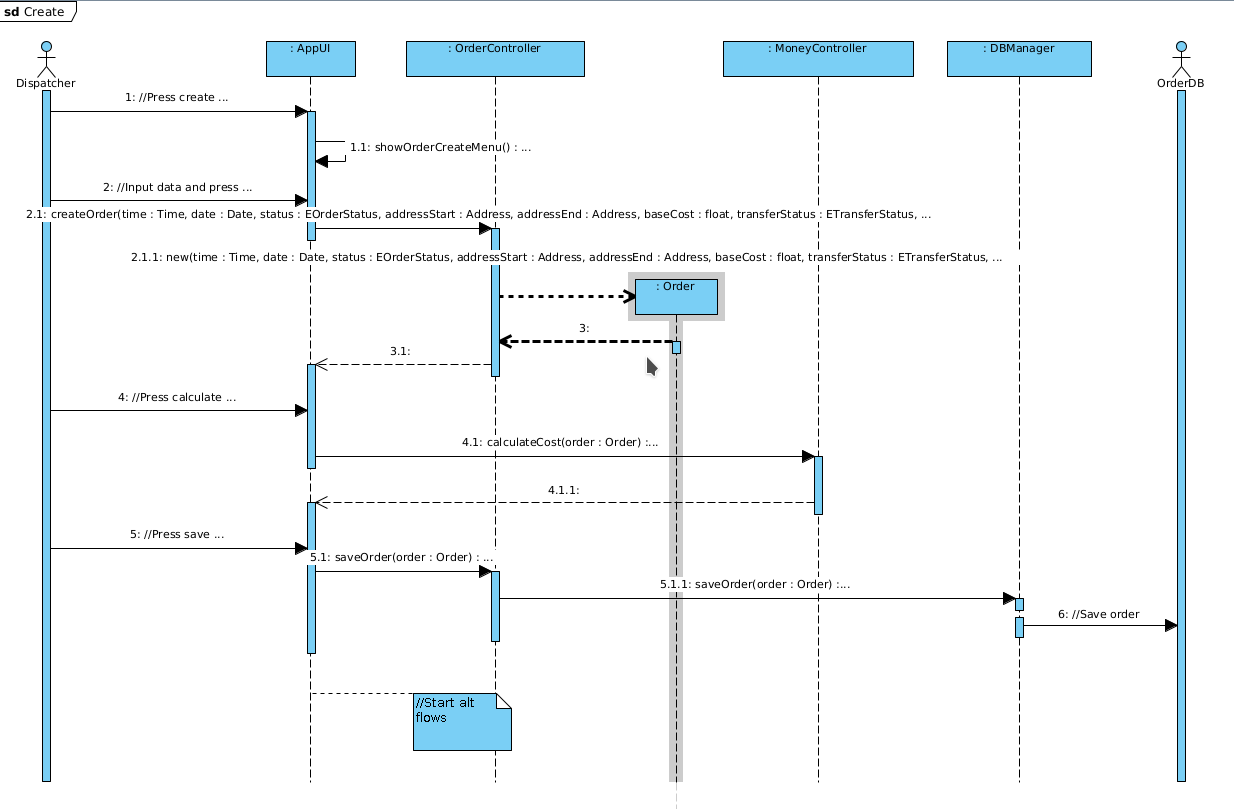


Диаграмма последовательности CRUD заказа (поток чтения):

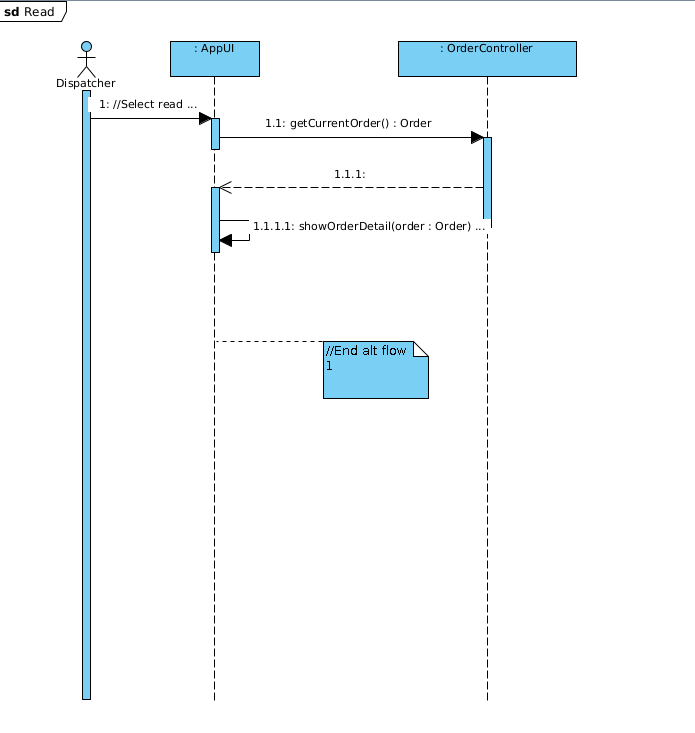


Диаграмма последовательности CRUD заказа (поток удаления заказа)

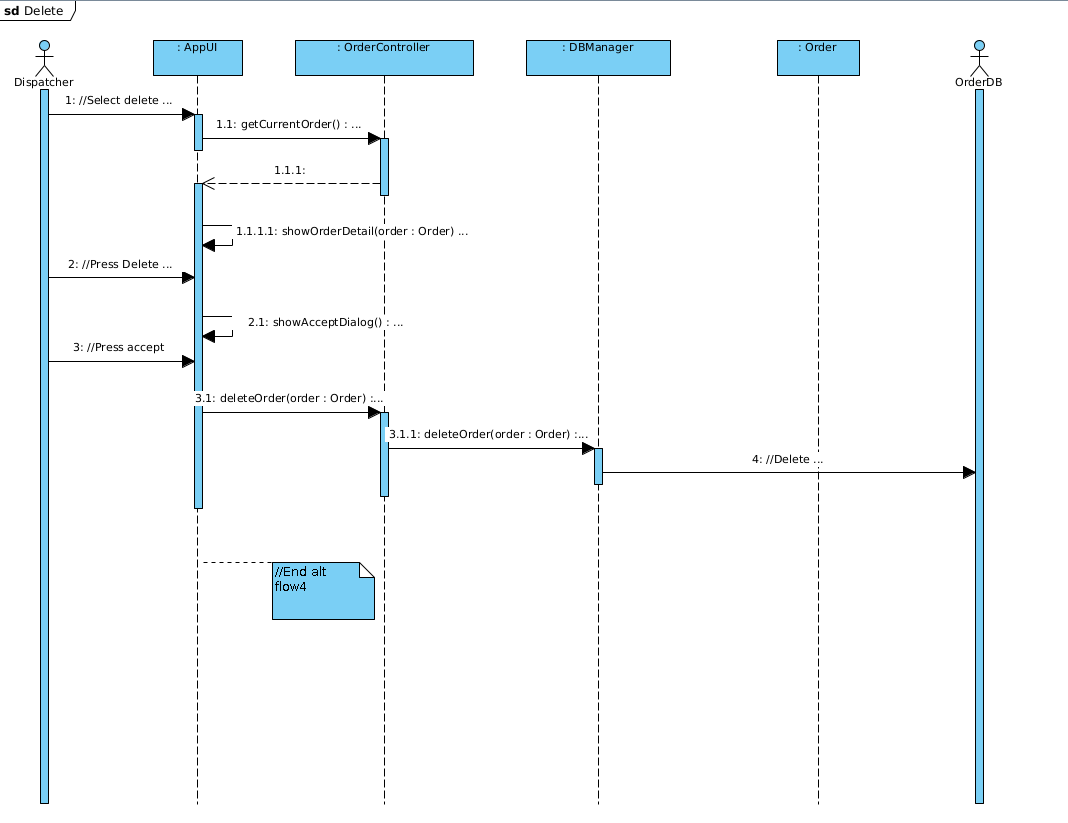


Диаграмма последовательности CRUD заказа (поток изменения данных заказа)

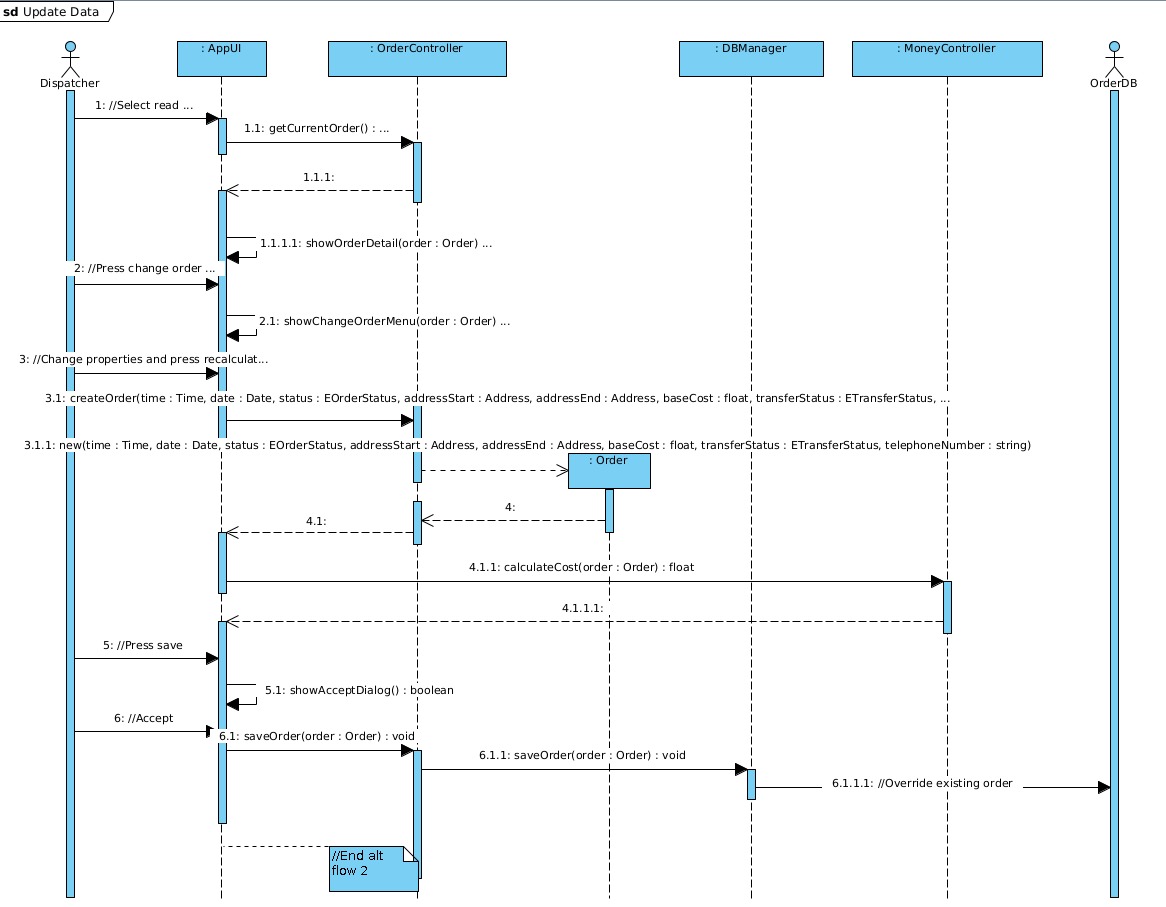


Диаграмма последовательности CRUD заказа (поток изменения водителя):

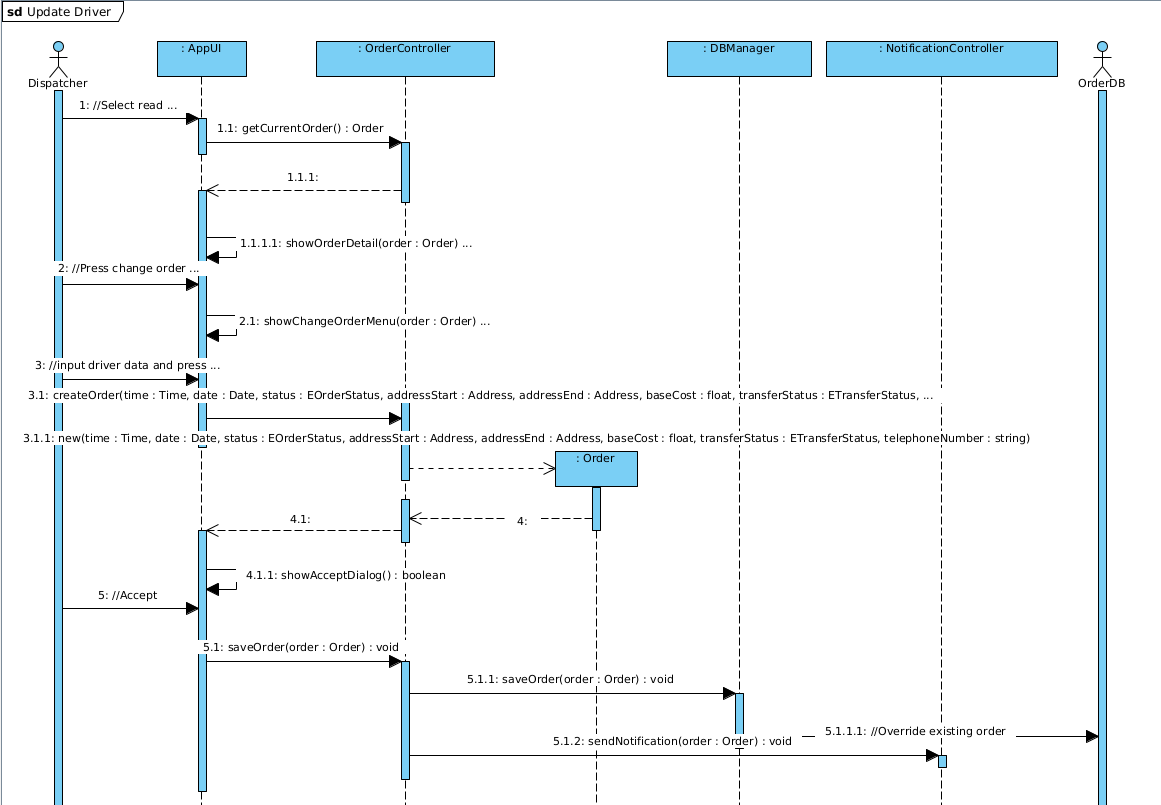


Диаграмма последовательности “Отчет о статусе поездки” (основной поток):

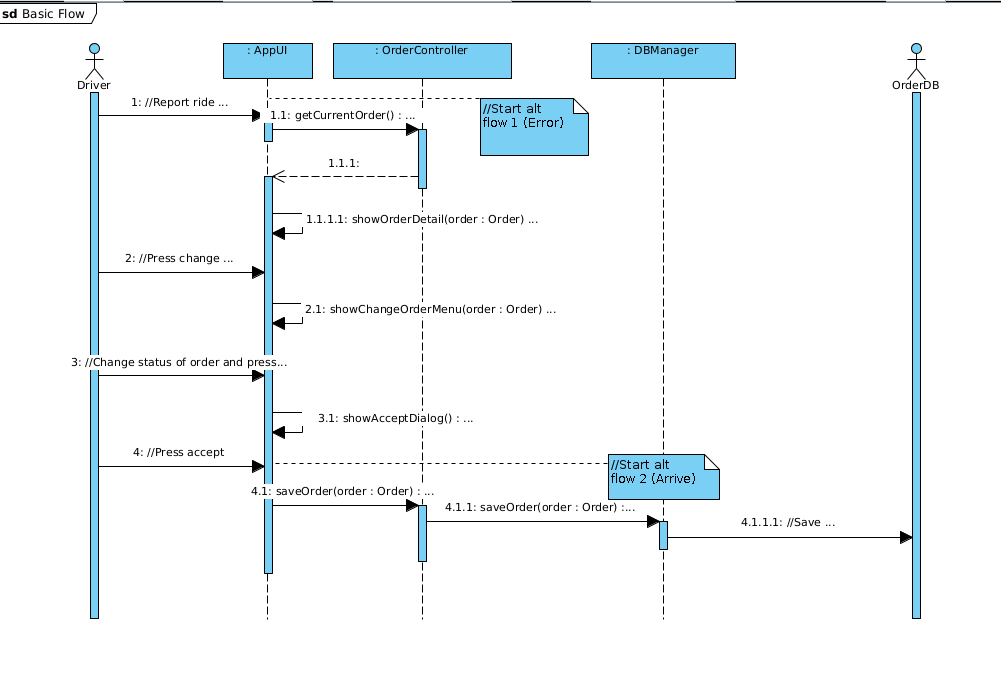


Диаграмма последовательности “Отчет о статусе поездки” (ошибка соединения):

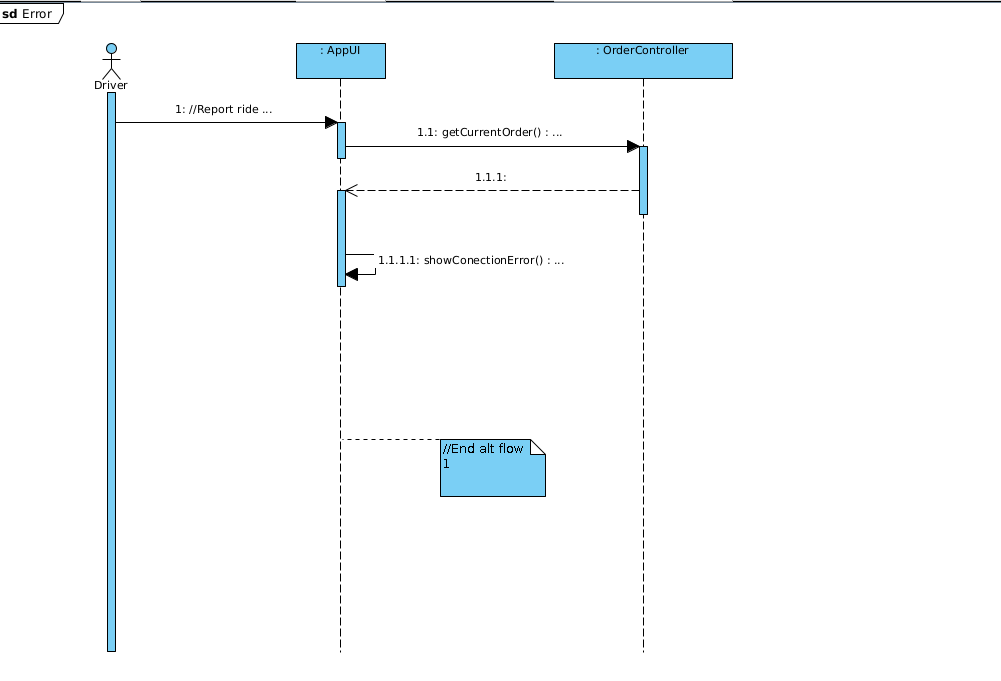


Диаграмма последовательности “Отчет о статусе поездки” (водитель прибыл к месту отправления):

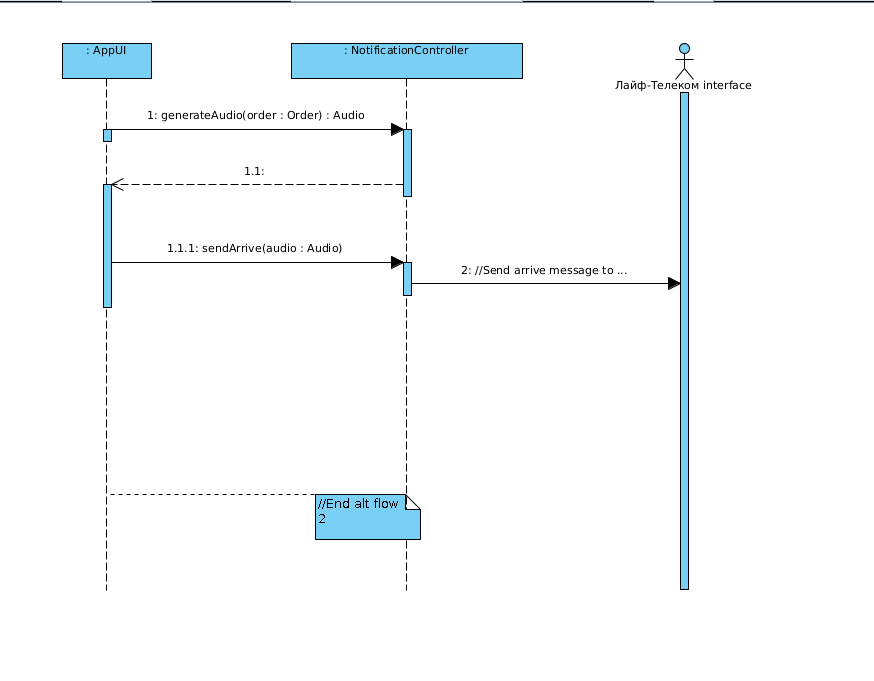


Диаграмма последовательности “Регистрация оплаты” (основной поток):

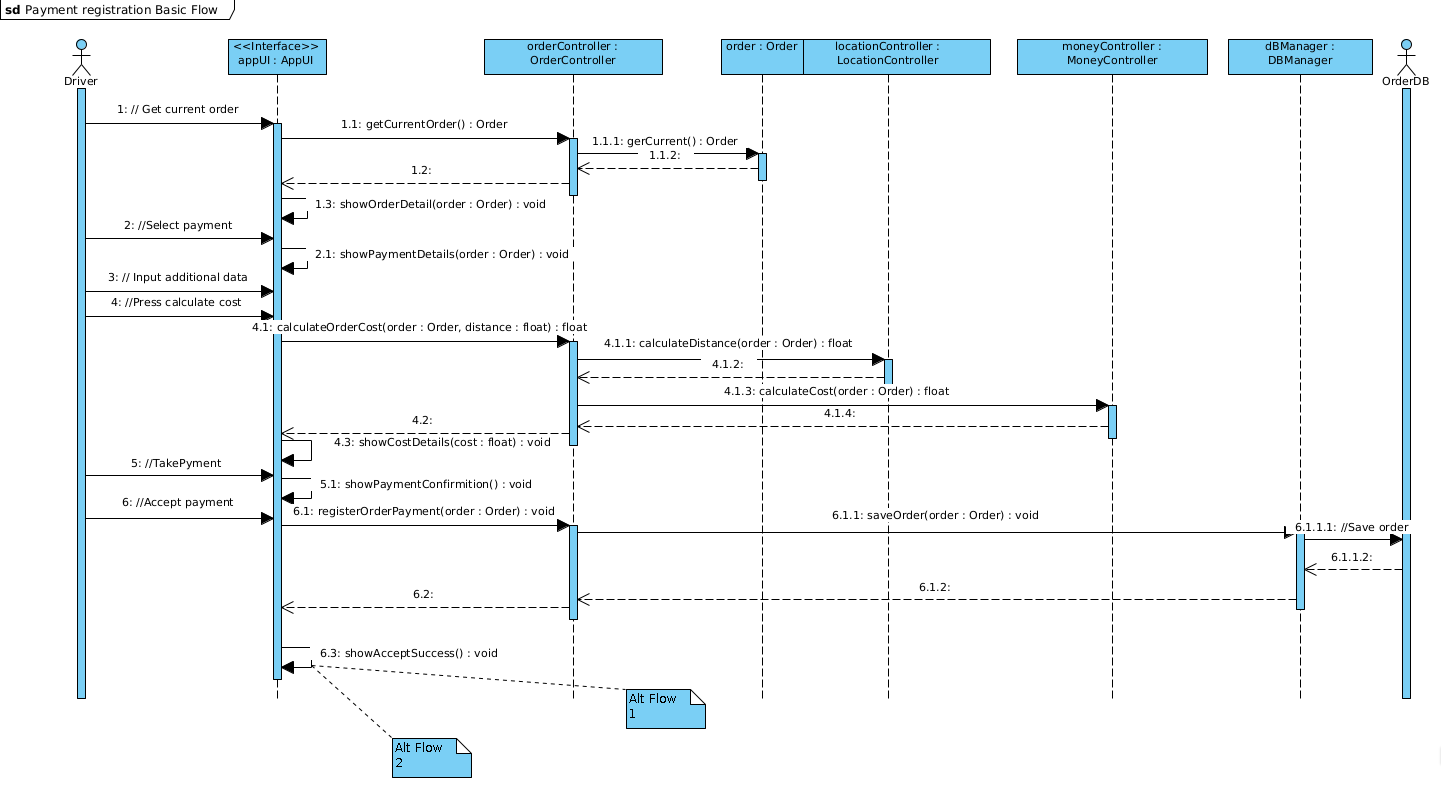


Диаграмма последовательности “Регистрация оплаты” (ошибка обработки данных сервером):

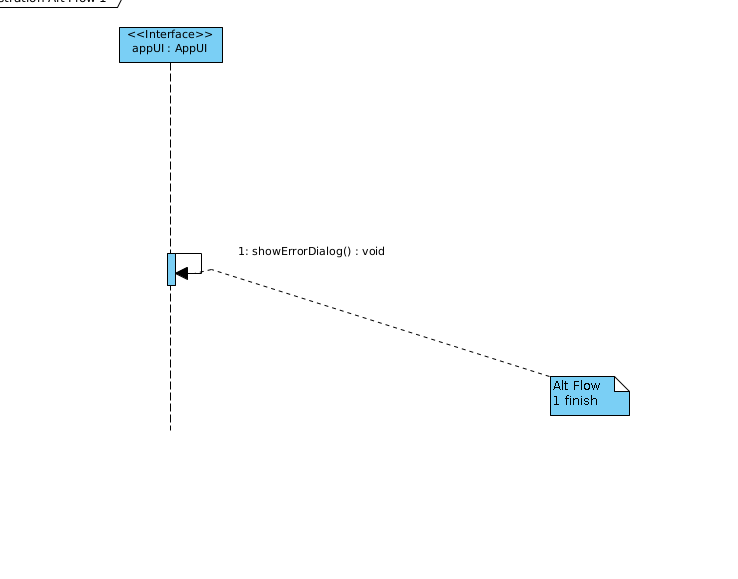


Диаграмма последовательности “Регистрация оплаты” (выписка чека):

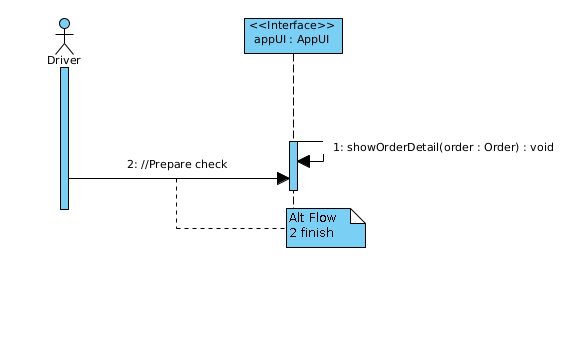


Диаграмма последовательности «Принять заказ» (основной поток):

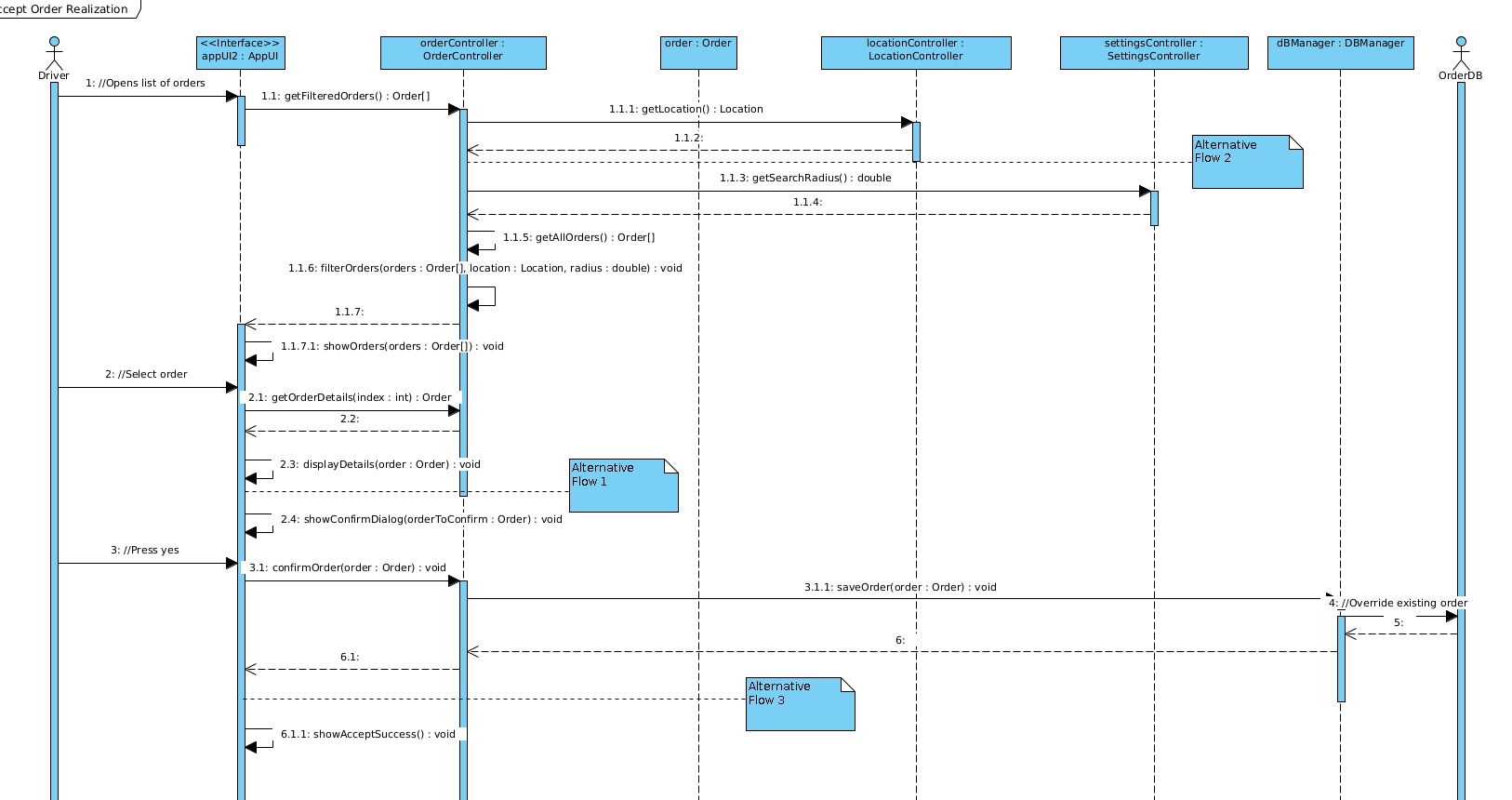


Диаграмма последовательности «Принять заказ» (отмена заказа на шаге подтверждения):

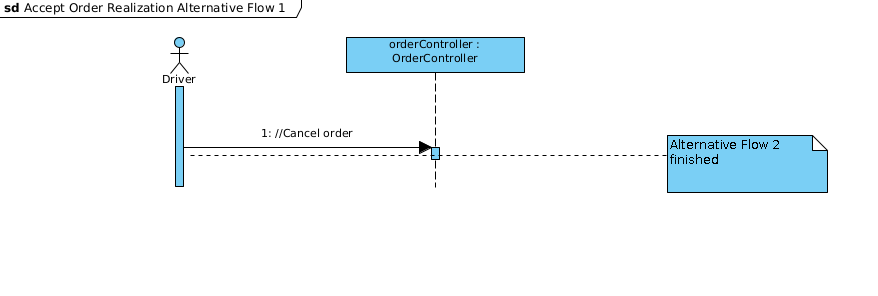


Диаграмма последовательности «Приянть заказ» (ошибка получения геоданных):

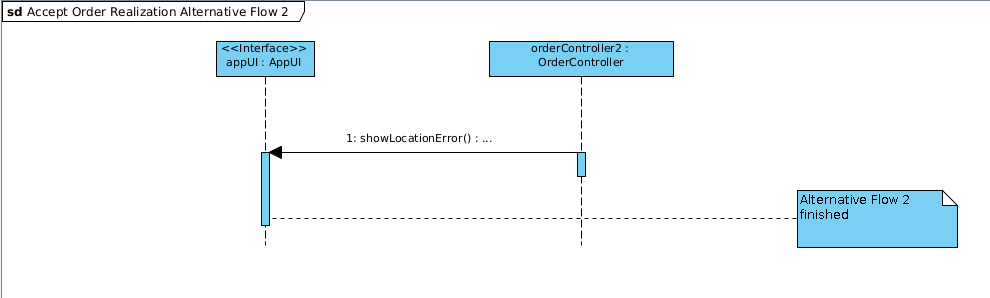
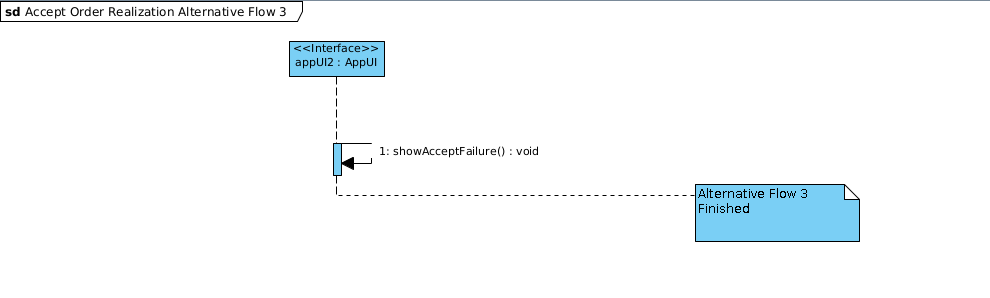


Диаграмма последовательности «Принять заказ» (ошибка конкурентности, когда заказу уже присвоен водитель):



## Физическое представление архитектуры

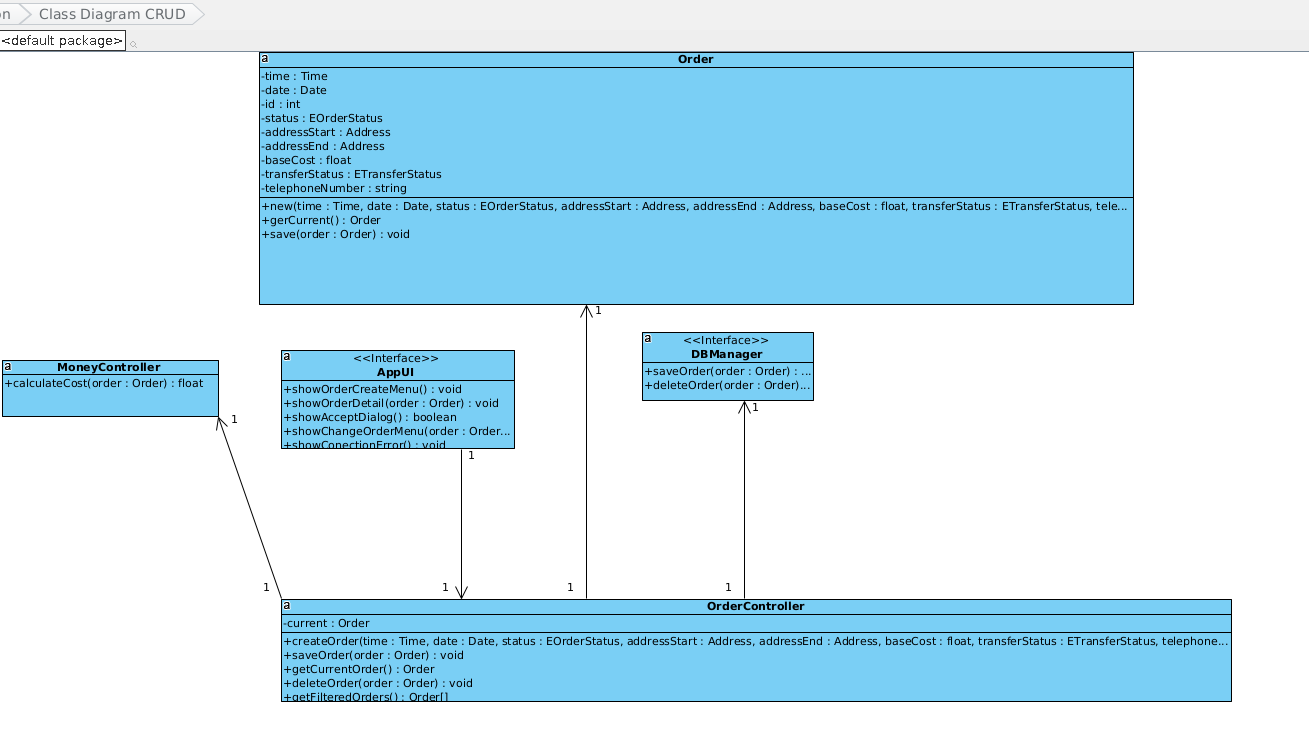
Графическая часть системы развёртывается на компьютерах в офисе таксопарка, мобильное приложение для водителей доступно для установки на смартфоны, базы данных хранятся на серверах.

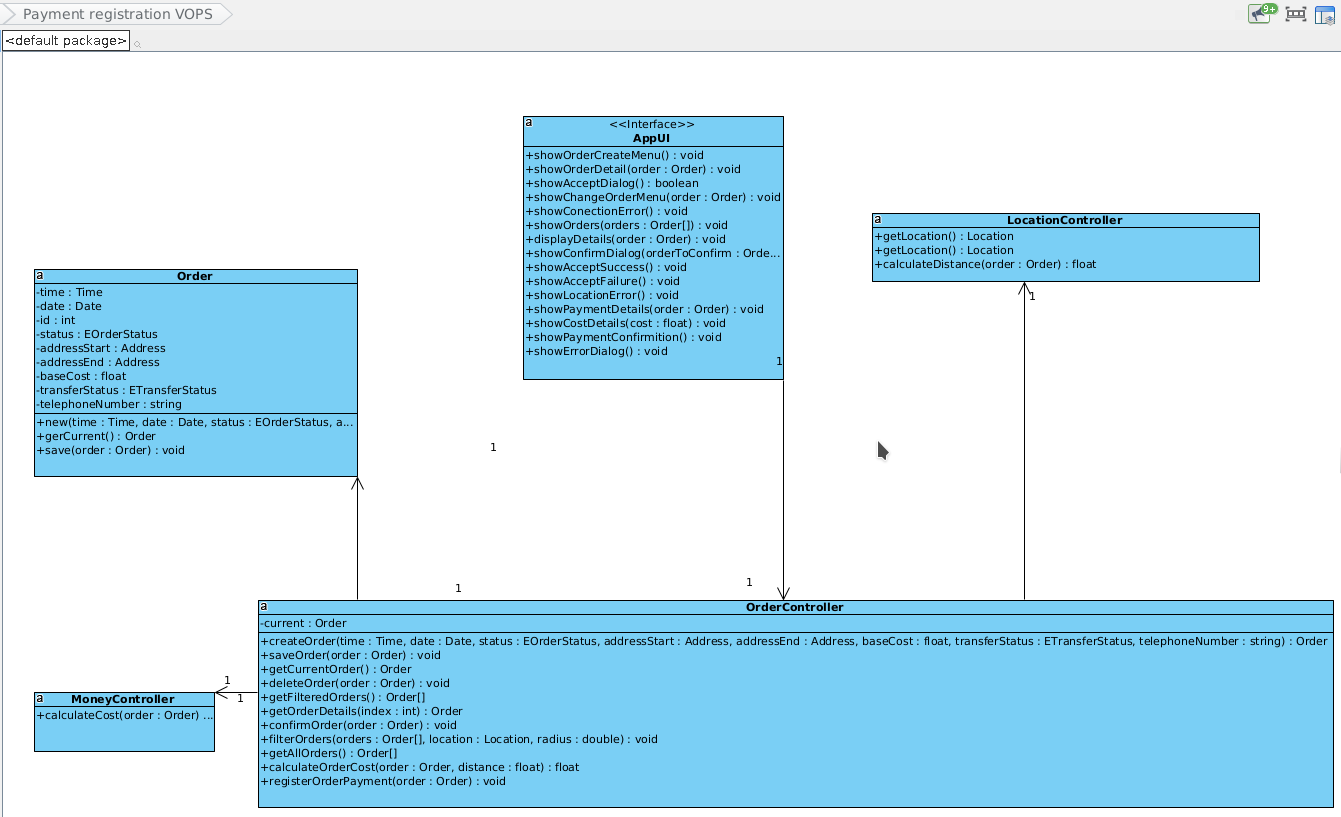
## Представление развертывания

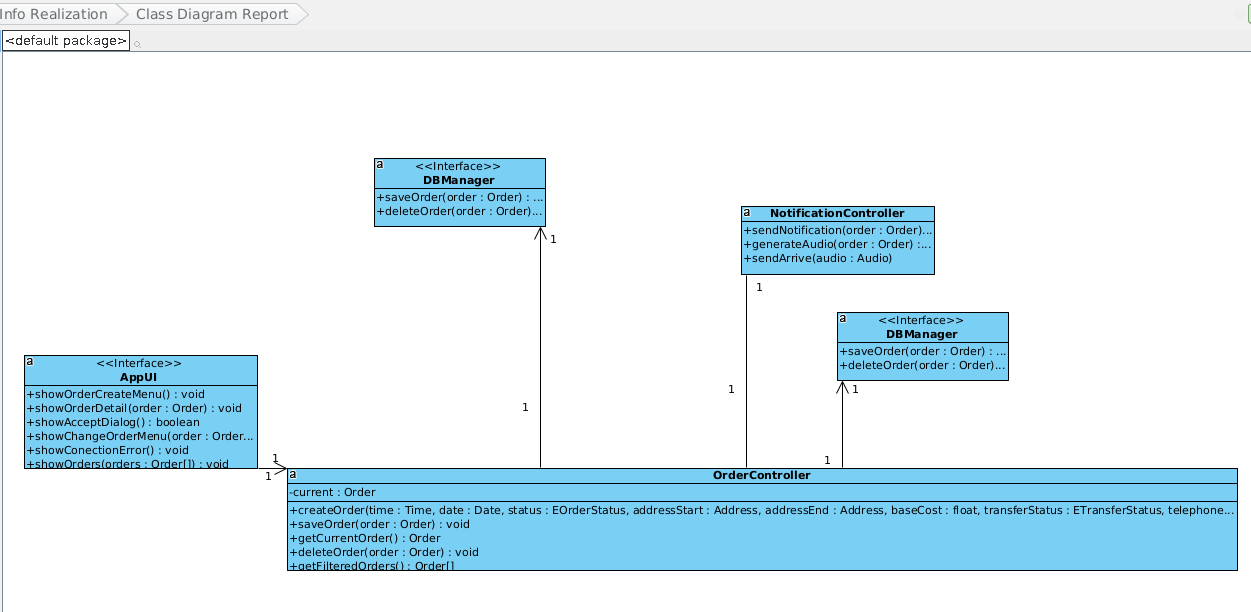
Система поставляется для установки на ПК в виде deb-пакетов или exe-файлов. Для мобильных устройств файлы в соответсвующем формате (apk, ipa) поставляются через магазин приложений.

## Представление архитектуры данных

## 







## Представление реализации

Серверная часть и часть для ПК реализована на Java и Kotlin, исполняемый код хранится в jar, dll/so файлах, файлы-ресурсы представлены в xml и json форматах. Мобильное приложение под Android - Kotlin, исполняемый код хранится в jar, файлы-ресурсы представлены в xml и json форматах; под IOS - Swift, файлы-ресурсы представлены в xml и json форматах.

## Представление разработки

В случае для серверной и ПК частей проект имеет стандартную для Java, Kotlin и Gradle структуру каталогов и файлов, для мобильных систем - соответсвующую им стандартную структуру. Разработка ведётся при помощи IntelliJ IDEA, Android Studio, XCode, сборка - Gradle 6 или XCode, тестирование юнит и интеграционными тестами. Используется система контроля версий git, проект хранится на сервере компании и доступен с помощью интерфейса GitLab.

## Нефункциональные аспекты

### Объем данных и производительность системы

Система поддерживает работу одновременно с 1800 пользователями и хранение данных о 550000 заказов в неделю.

### Гарантии качества работы системы

Системная архитектура стабильно работает при указанных выше нагрузках, может самостоятельно восстановиться при возникновении сбоя.

# Приложения

## Словарь терминов

ИС - Информационная система

Адрес подачи - адрес места отправления, начала поездки

Статус поездки - «Прибыл к адресу подачи», «В пути» или «Поездка окончена»