МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе
по дисциплине «Спецификация, проектирование и архитектура программных систем»

Студент гр. 0304 _____ Крицын Д.Р.
Студент гр. 0304 _____ Голиков А.В.
Студент гр. 0304 _____ Решоткин А.С.
Студент гр. 0304 _____ Докучаев Р.А.
Студент гр. 0304 _____ Козиков А.Е.
Преподаватель _____ Романенко С.А.

Санкт-Петербург
2022

ВВЕДЕНИЕ.

Работа представляет из себя проект по автоматизации системы контроля транспорта. Для системы изначально было разработано ТЗ, в соответствии с которым разрабатывалась архитектура системы и описывалась предметная область. Результат работы представлен в данной пояснительной записке.

1. ОБЩИЕ ПОЛОЖЕНИЯ

1.1. Полное наименование системы

Веб-приложение с модулями для сотрудников депо и пассажиров. Условное обозначение: ИС СКТ.

1.2. Цели, назначение и области использования АС

Система предназначена для распределения автобусов, троллейбусов, трамваев по маршрутам, контроль местоположения транспортных средств, информирование пассажиров о времени прибытия, системы валидации для проверки оплаты проезда банковскими картами и проездными.

Цели создания системы:

- Организовать базу данных с расписанием транспортных средств.
- Предоставить работникам депо инструменты для создания и редактирования расписания.
- Создать инструмент отслеживания ТС на маршруте.
- Создать надежную систему оплаты проезда. И возможность проверки оплаченного проезда.
- Разработать инструмент отслеживания и отправки уведомлений о прибытии ТС для пользователя.
- Разработать веб-приложение, объединяющее созданные инструменты, с предоставлением её пользователям.

2. ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

2.1. Описание предметной области в нотации IDEF0

2.1.1. Словарь терминов предметной области

1. **Маршрут следования** - заранее определенный маршрут передвижения транспортных средств, по которому они постоянно совершают поездки из одного конца в другой.
2. **Остановочный пункт** - место, где транспортные средства, следующие по маршруту, совершают остановку с целью обеспечения высадки / посадки пассажиров.
3. **Расписание маршрута** - примерное описание времени прибытия транспортного средства на каждую остановку в маршруте следования.
4. **Проездной билет** - документ, удостоверяющий наличие права пассажира на проезд на определенном транспортном средстве.
5. **Личный счет** - денежный счет, привязанный к учетной записи пользователя системы. Личный счет можно пополнить через банковскую систему, деньги на личном счету можно потратить на покупку билетов и/или пополнение проездного, или вывести.
6. **Сотрудник технической поддержки** - сотрудник, отслеживающий стабильность работы веб-приложения и входящие отчёты об ошибках, отправленные пользователями приложения. В случае неисправностей сотрудник технической поддержки может связаться с разработчиками приложения для устранения неисправностей.
7. **Веб-модуль** - один из интерфейсов веб-приложения, обеспечивающий доступ к части его функционала.
8. **Пользователь** - лицо, выступающее в роли пассажира или диспетчера в данной автоматизированной системе. В зависимости от роли пользователь имеет доступ к разным частям функционала приложения.

2.1.2. Контекстная диаграмма IDEF0

На рисунке 1 представлена контекстная диаграмма.

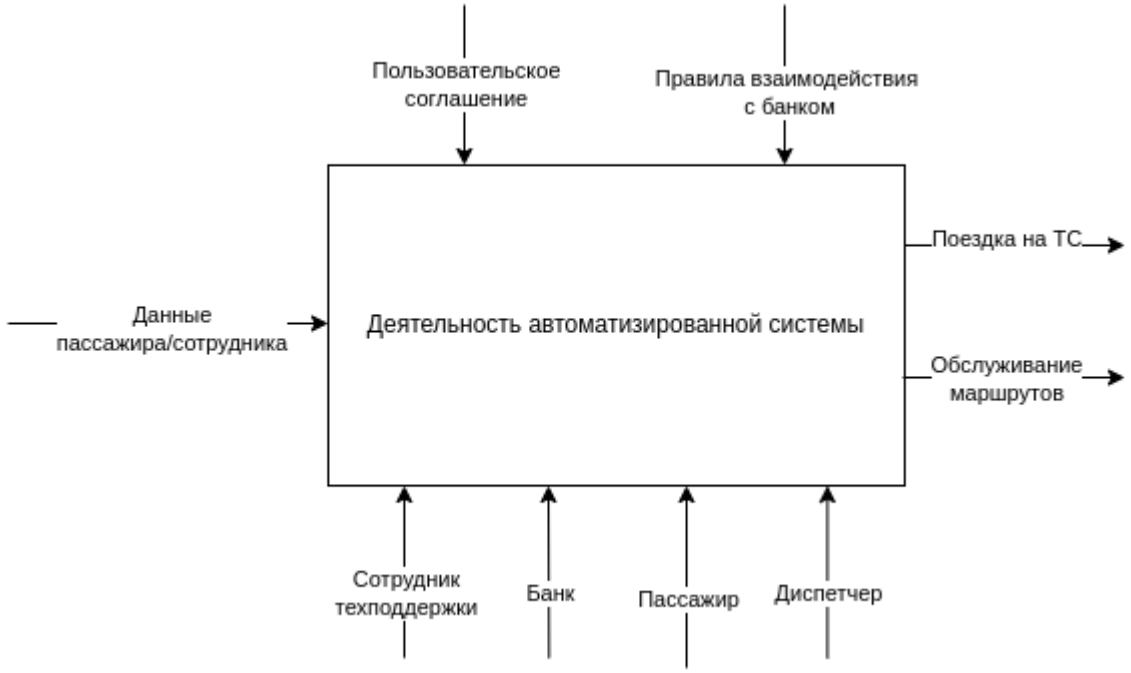

Рисунок 1. Контекстная диаграмма.

На данной диаграмме определены границы системы. Представленная система имеет следующие механизмы: Сотрудник техподдержки, Банк, ПО, отслеживающее местоположение транспорта, СУБД маршрута.

Так как система подразумевает личный кабинет пользователя/сотрудника, то для входа необходим аккаунт в системе. Для взаимодействия с системой предусмотрены элементы управления: пользовательское соглашение, правила взаимодействия с банком.

2.1.3. Диаграмма IDEF0 1-го уровня

На рисунке 2 отображена диаграмма IDEF0 1-го уровня.

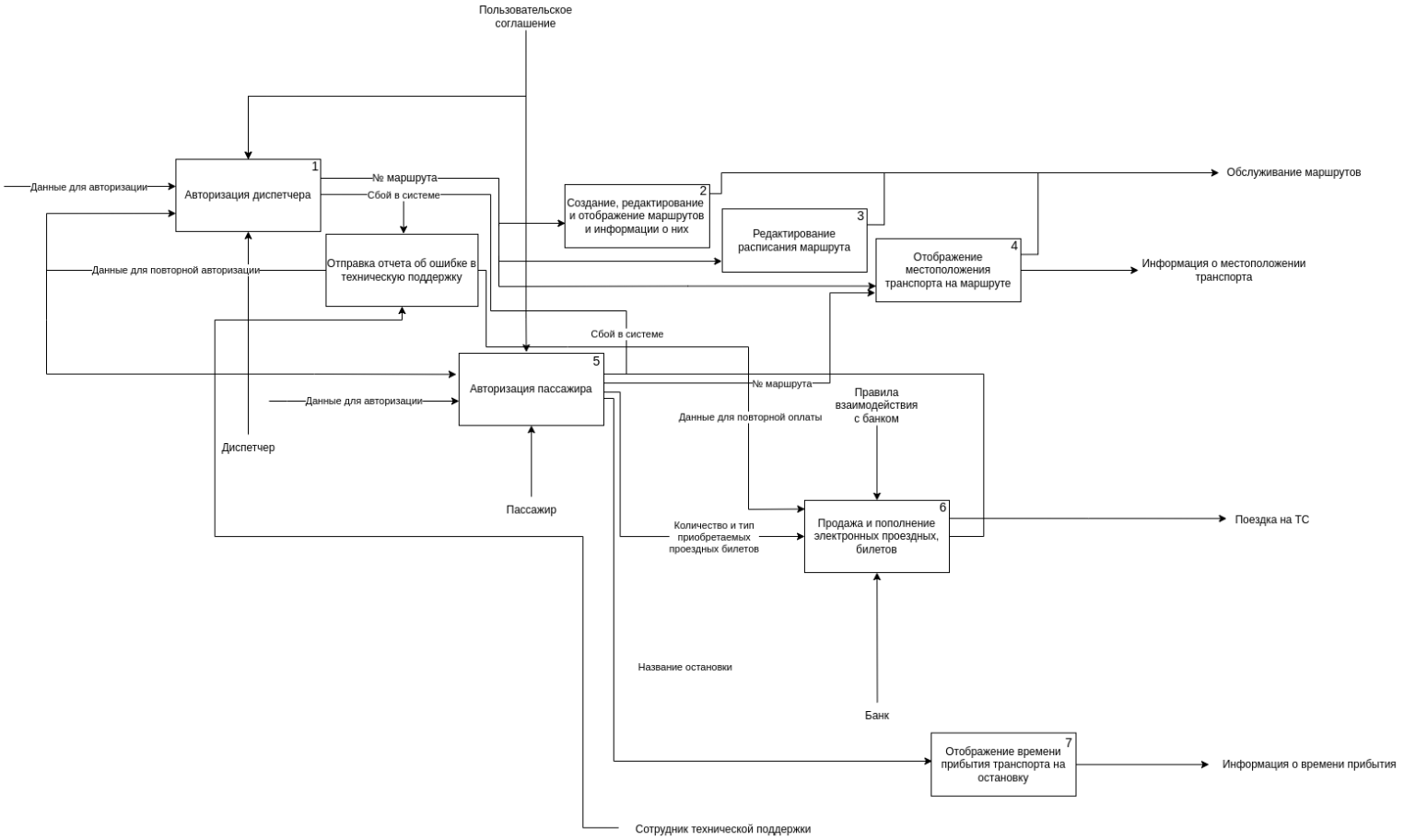

Рисунок 2. Диаграмма IDEF0 1-го уровня A0

1) Блок “Авторизация пассажира” включает в себя функцию авторизации пассажира в системе. На вход блок получает данные для авторизации пользователя, а на выходе возвращает название остановки, № маршрута, количество и тип приобретаемых проездных билетов или, в случае системного сбоя, управление переходит к блоку “Отправка отчёта об ошибке в техническую поддержку”, после которого есть возможность вновь авторизоваться. В качестве управляющего элемента блок получает пользовательское соглашение.
2) Блок “Отображение местоположения транспорта на маршруте” включает в себя функцию отображения местоположения

транспорта на маршруте. На вход блок получает информацию о номере маршрута, а на выходе возвращает информацию о маршруте. В качестве механизма блок получает данные ПО, отслеживающего местоположение транспорта.

3) Блок “Отображение времени прибытия транспорта на остановку” включает в себя функцию отображения времени прибытия транспорта на остановку. На вход блок получает название остановки, а на выходе возвращает информацию о времени прибытия ТС на остановку.
4) Блок “Продажа и пополнение электронных проездных, билетов” включает в себя функцию финансовых операций с электронными проездными и билетами: продажа, пополнение. На вход блок получает количество и тип приобретаемых проездных билетов, а на выходе возвращает поездку на ТС или, в случае системного сбоя, управление переходит к блоку “Отправка отчёта об ошибке в техническую поддержку”, после которого есть повторная попытка оплаты. В качестве управляющего элемента блок получает правила взаимодействия с банком. В качестве механизма блок получает данные из банка.
5) Блок “Авторизация диспетчера” включает в себя авторизацию сотрудника системы контроля общественного транспорта. На вход получает данные для авторизации, на выходе возвращает № маршрута или, в случае системного сбоя, управление переходит к блоку “Отправка отчёта об ошибке в техническую поддержку”, после которого есть возможность вновь авторизоваться. В качестве управляющего элемента блок получает пользовательское соглашение.

6) Блок “Создание, редактирование и отображение маршрутов и информации о них” включает в себя функцию создания нового маршрута в системе контроля транспорта диспетчером, а также последующее изменение характеристик маршрута тем же диспетчером. На вход получает № маршрута, на выходе вместе с блоками “Редактирование расписания маршрутов” и “Отображение местоположения транспорта на маршруте” обеспечивает возможность обслуживания маршрута.
7) Блок “Редактирование расписания маршрутов” включает в себя функцию изменения текущего графика движения ТС на маршрутах. На вход получает № маршрута, на выходе вместе с блоками “Создание, редактирование и отображение маршрутов и информации о них” и “Отображение местоположения транспорта на маршруте” обеспечивает возможность обслуживания маршрута.
8) Блок “Отображение местоположения транспорта на маршруте” включает в себя функцию контроля за текущим положением транспорта диспетчеру. На вход получает № маршрута, на выходе вместе с блоками “Создание, редактирование и отображение маршрутов и информации о них” и “Редактирование расписания маршрутов” обеспечивает возможность обслуживания маршрута диспетчером

2.1.4. Диаграммы IDEF0 2-го уровня

На рисунке 3 отображена диаграмма IDEF0 2-го уровня A1.

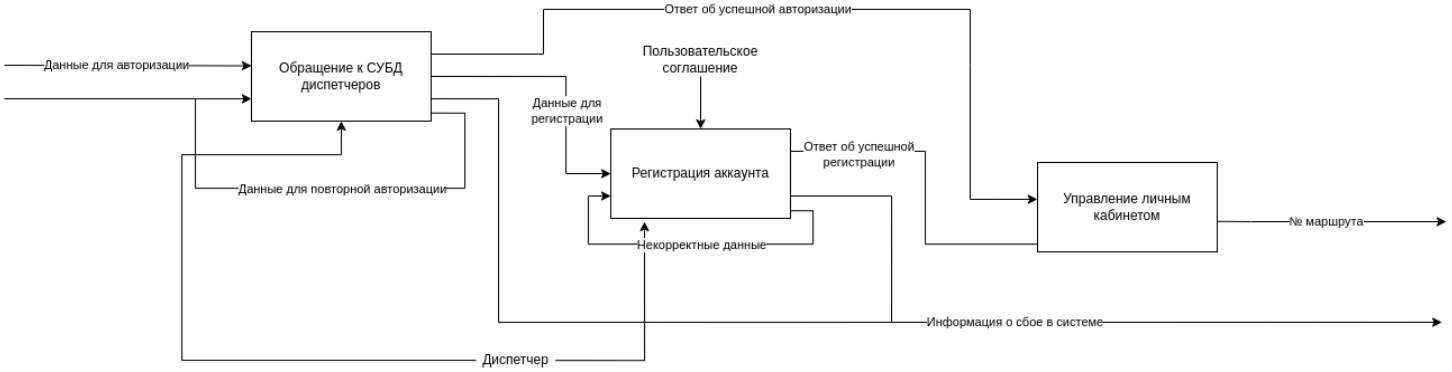

Рисунок 3. Диаграмма IDEF0 2-го уровня A1.

Данная диаграмма описывает процесс авторизации диспетчера в системе. При регистрации учетной записи диспетчера проводится проверка данных для авторизации в СУБД диспетчеров, чтобы предотвратить регистрацию сторонних пользователей. При входе в аккаунт такая проверка не производится. После успешного входа или регистрации диспетчер в системе может изменять различные параметры маршрутов через личный кабинет. При неудачной регистрации или входе в аккаунт предусмотрена возможность повторной попытки регистрации или входа.

На рисунке 4 отображена диаграмма IDEF0 2-го уровня A2.

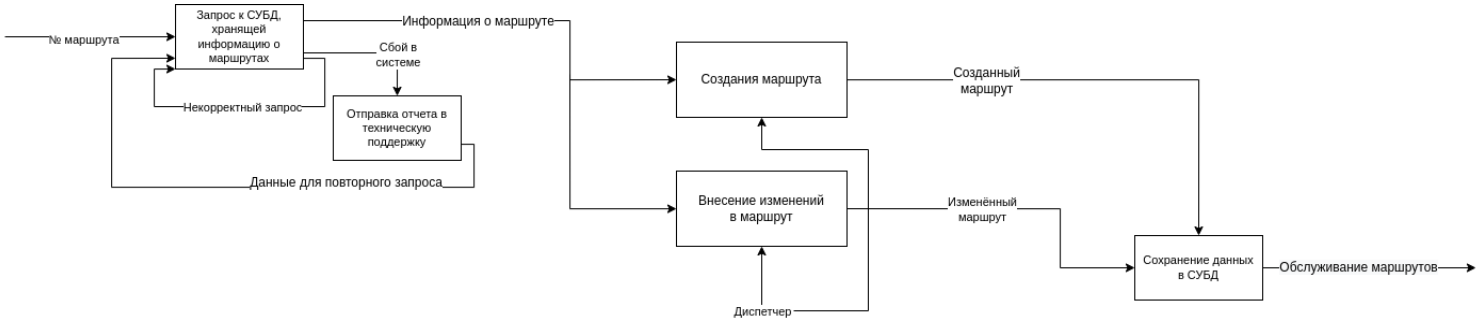

Рисунок 4. Диаграмма IDEF0 2-го уровня A2

Данная диаграмма детализирует процесс создания, редактирования маршрута и его отображения. Диспетчер передает номер маршрута, после чего из СУБД происходит загрузка информации о маршруте. При сбое или некорректном запросе происходит повторная попытка получить информацию о маршруте из СУБД или обращение к службе поддержки. Если информация о маршруте получена успешно, оно загружается в ПО. ПО предоставляет сотруднику депо необходимые инструменты для создания маршрута. После того, как все необходимые данные для создания маршрута были добавлены новый маршрут сохраняется в СУБД. Также ПО предоставляет пользователю возможность редактирования уже существующего маршрута, который также после редактирования обновляется в СУБД. При успешном сохранении мы реализуем процесс обслуживания маршрута.

На рисунке 5 отображена диаграмма IDEF0 2-го уровня А3.

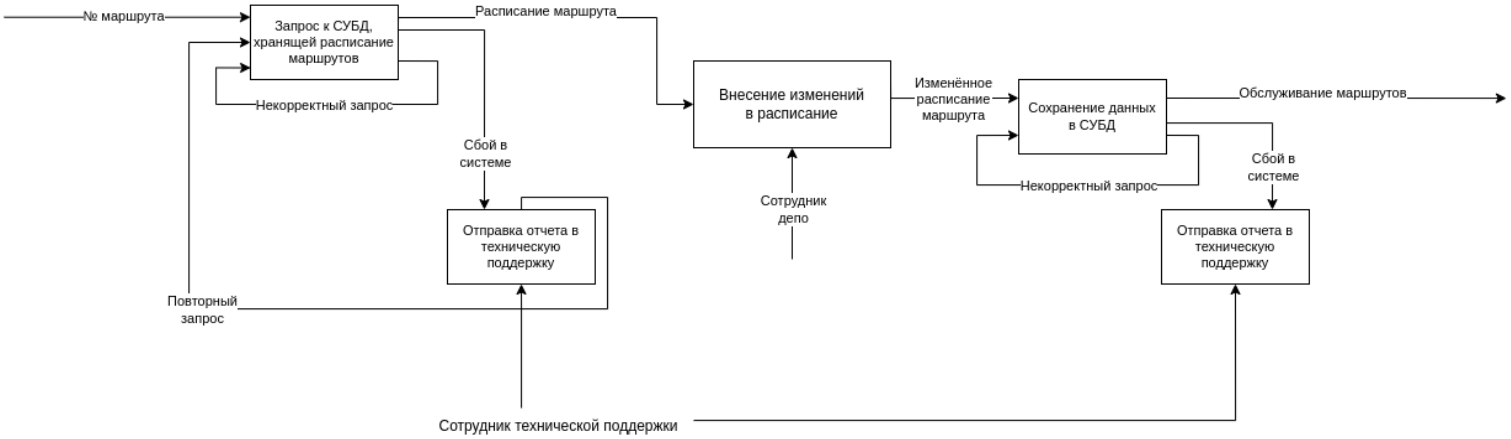

Рисунок 5. Диаграмма IDEF0 2-го уровня А3

Данная диаграмма детализирует процесс редактирования расписания маршрута. Пользователь передает номер маршрута, после чего из СУБД происходит загрузка информации о маршруте. При сбое или некорректном запросе происходит повторная попытка получить расписание из СУБД или обращение к службе поддержки. Если расписание получено успешно, оно загружается в ПО. ПО предоставляет сотруднику депо необходимые инструменты для редактирования расписания маршрута и сохранения его в СУБД. В случае ошибки сохранения производится повторной запрос к СУБД или обращение к службе поддержки. При успешном сохранении мы реализуем процесс обслуживания маршрута.

На рисунке 6 отображена диаграмма IDEF0 2-го уровня A4.

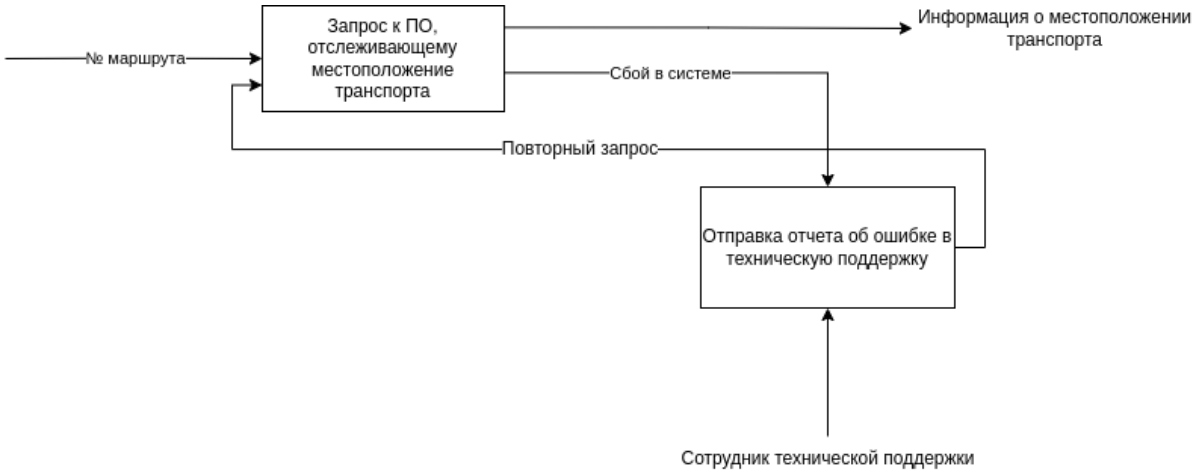

Рисунок 6. Диаграмма IDEF0 2-го уровня A4

Данная диаграмма детализирует процесс отслеживания местоположения транспорта на маршруте. Пользователь передает номер маршрута. Далее формируется запрос к ПО, отслеживающему местоположение транспорта. После успешного получения запроса пользователю отправляется информация о местоположении транспорта. В случае системного сбоя происходит отправка отчета в техническую поддержку. Далее происходит повторный запрос. Для блока “Запрос к ПО, отслеживающему местоположение транспорта” используется механизм ПО, отслеживающее местоположение транспорта. Для блока “Отправка отчета в техническую поддержку” используется механизм сотрудник технической поддержки.

На рисунке 7 отображена диаграмма IDEF0 2-го уровня A5.

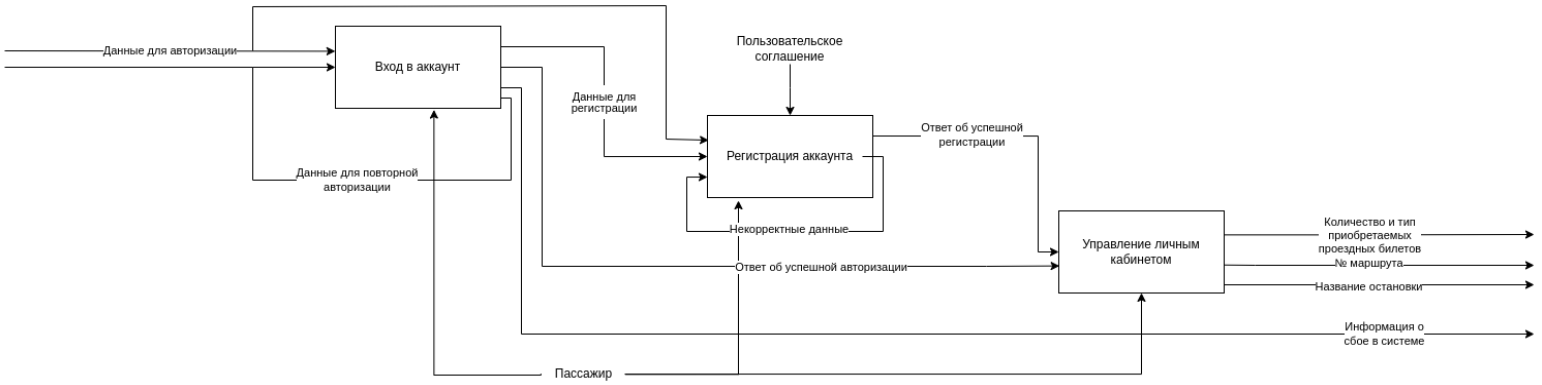

Рисунок 7. Диаграмма IDEF0 2-го уровня A5.

Данная диаграмма детализирует процесс авторизации пассажира в системе. Пользователь может сразу зарегистрироваться в системе, а может начать процесс регистрации после неуспешной попытки входа в систему, при этом попытку регистрации или входа в учетную запись можно повторить при вводе некорректных данных. После успешного входа или регистрации пассажир через личный кабинет может приобретать проездные билеты, просматривать местоположение транспорта и примерное время прибытия на остановку.

На рисунке 8 отображена диаграмма IDEF0 2-го уровня А6.

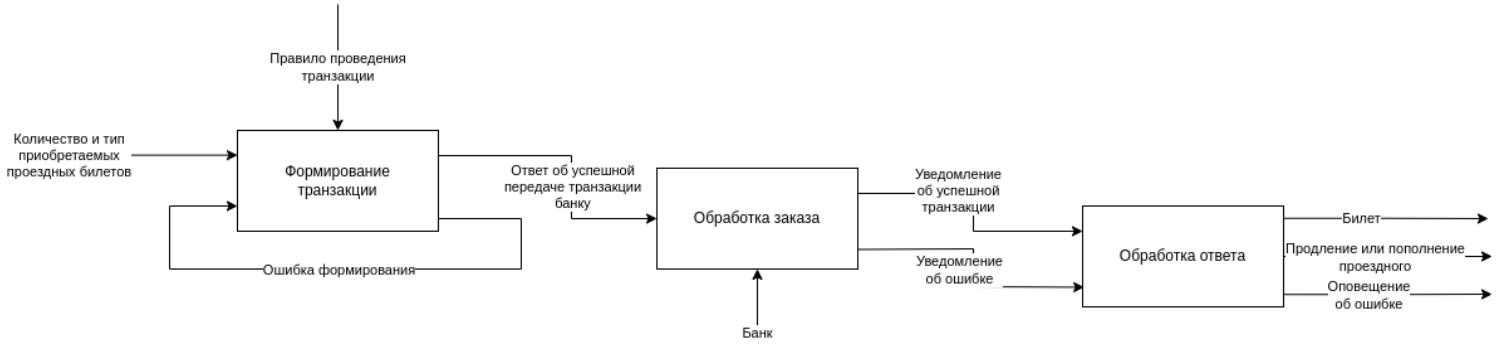

Рисунок 8. Диаграмма IDEF0 2-го уровня А6

Данная диаграмма описывает процесс покупки или пополнения проездного, а также электронных билетов. Как можно видеть, изначально управление передается блоку “формирование транзакции”, после чего, на выходе получается транзакция, передающаяся банку. Также предусмотрено повторное формирование транзакции, если она была сформирована некорректно. Управляющий элемент - правила проведения транзакции. Далее управление передается блоку “Обработка заказа” с механизмом - Банк. После обработки заказа банком управление передается блоку “Обработка ответа”, и он, в зависимости от успешной или неуспешной обработки заказа может выдать билет, пополнить или продлить проездной, либо, в случае ошибки при обработке, выдать извещение об ошибке. В данном блоке один механизм - ПО.

На рисунке 9 отображена диаграмма IDEF0 2-го уровня A7.

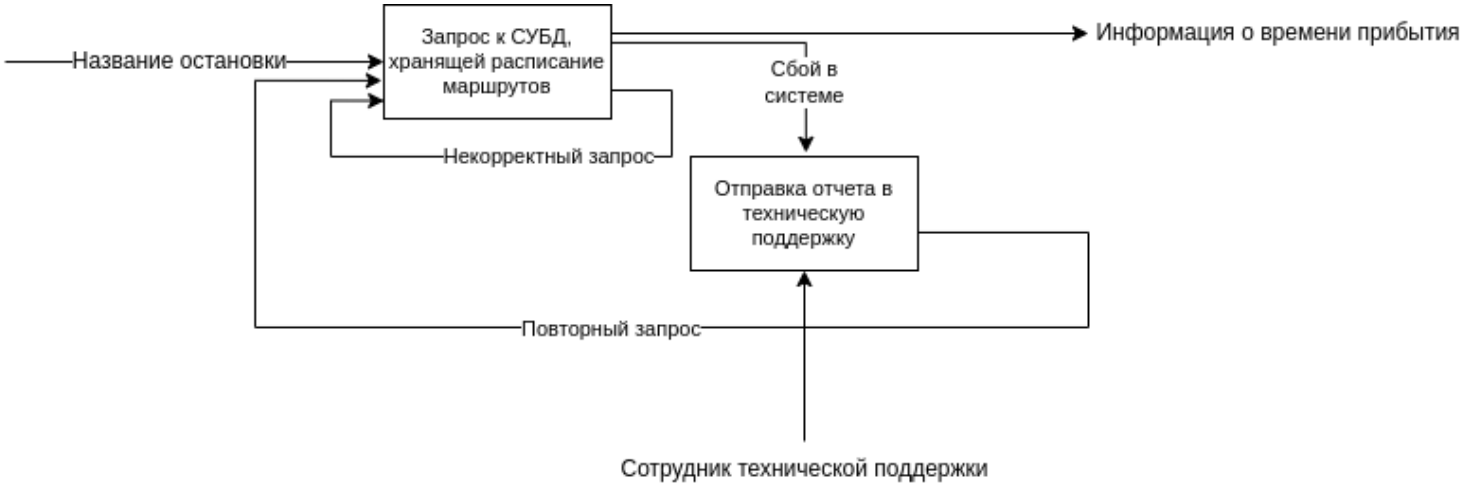

Рисунок 9. Диаграмма IDEF0 2-го уровня A7

Данная диаграмма детализирует процесс отображения информации о времени прибытия транспорта на остановку. Пользователь передает информацию о названии остановки, которая ищется в СУБД, хранящей расписание маршрутов. После успешного получения запроса пользователю отправляется информация о времени прибытия. В случае системного сбоя происходит отправка отчета в техническую поддержку. Далее происходит повторный запрос к СУБД. В ином случае, если запрос пользователя оказался некорректным, то также происходит повторный запрос к СУБД.

2.2. Описание потоков данных

2.2.1. Диаграмма потоков данных

На рисунке 10 представлена диаграмма потоков данных.

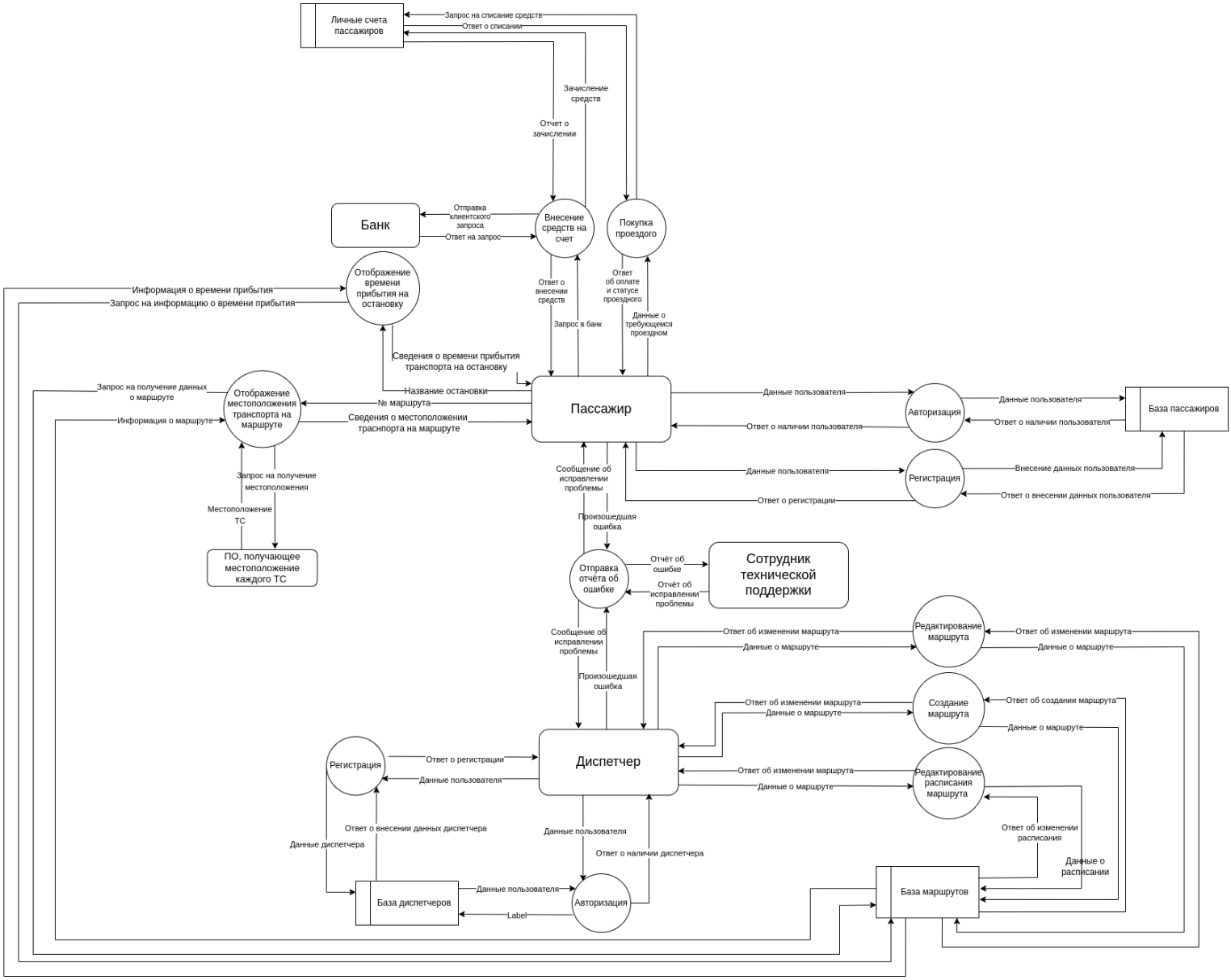

Рисунок 10. Диаграмма потоков данных

Диаграмма потоков описывает все процессы и связанные с ними сущности в соответствии с техническим заданием, которые необходимы для корректной работы системы. На данной диаграмме представлены сущности основных пользователей системы (пассажир, диспетчер, сотрудник технической поддержки), а также сущность банка.

2.3. Модель предметной области

2.2.1. Диаграмма отношений сущностей

На рисунке 11 представлена диаграмма отношений сущностей.

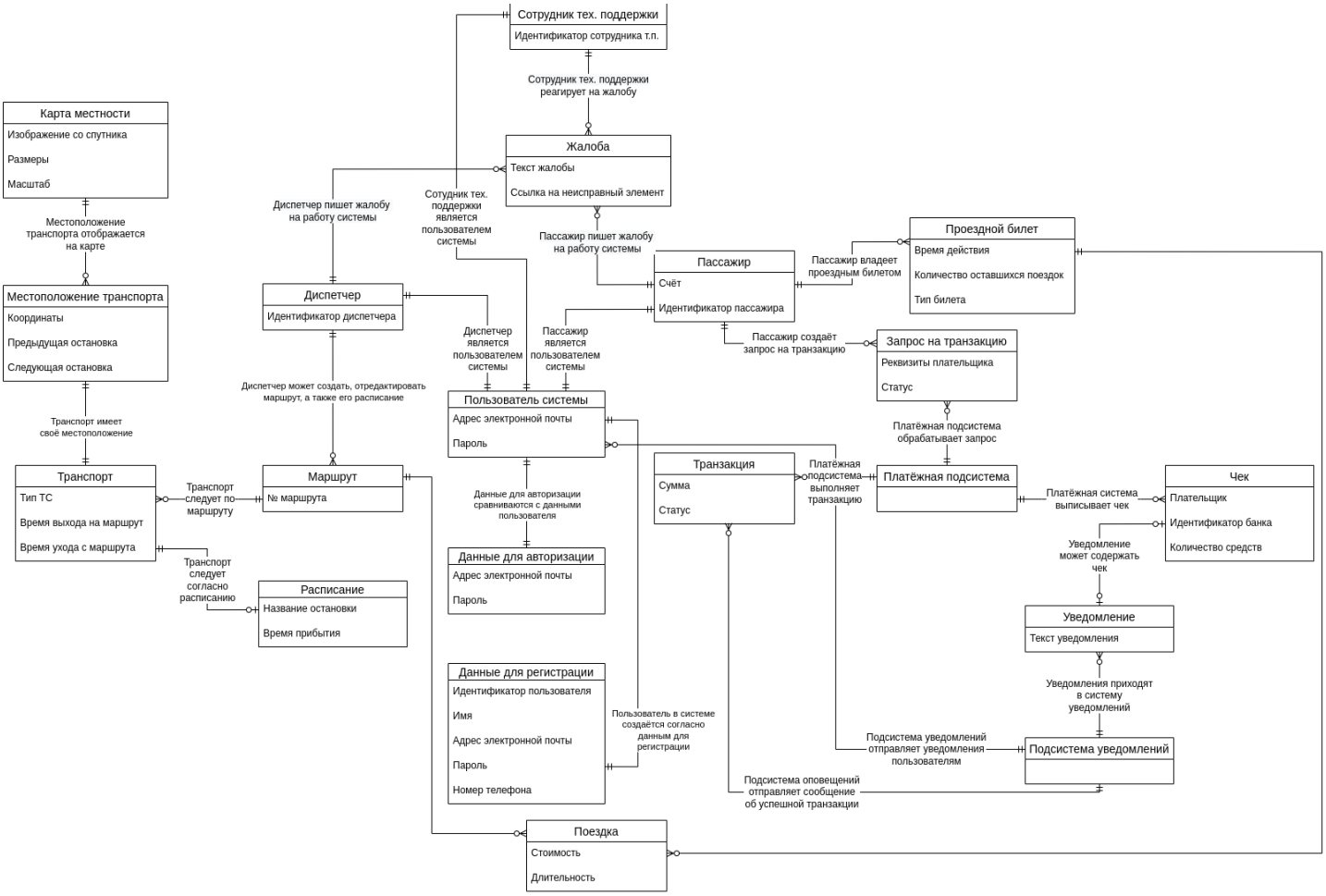

Рисунок 11. Диаграмма отношений сущностей

Данная диаграмма описывает отношения между следующими сущностями:

- **Данные для авторизации** хранят адрес электронной почты и пароль пользователя, и используются для авторизации пользователя в системе.
- **Данные для регистрации** хранят идентификатор пользователя, его имя, адрес электронной почты, пароль и номер телефона, и используются для создания нового пользователя в системе.

- **Пользователь системы** хранит свой адрес электронной почты и пароль - данные, необходимые для авторизации пользователя в системе.
- **Пассажир** хранит свой личный счёт и идентификатор, с помощью которого осуществляются различные операции (пополнение личного счёта, приобретение проездного билета, подача жалобы). Пассажир может владеть несколькими проездными билетами.
- **Диспетчер** хранит свой идентификатор, с помощью которого осуществляются операции по созданию и редактированию маршрутов и расписания. Директор, как и пассажир, тоже может подать жалобу на работу системы.
- **Сотрудник технической поддержки** хранит свой идентификатор, и может отвечать на жалобы других пользователей на работу приложения.
- **Транзакция** хранит сумму транзакции и текущий статус выполнения транзакции. Транзакция выполняется платёжной подсистемой.
- **Запрос на транзакцию** хранит реквизиты создавшего запрос плательщика, и статус запроса. Запрос обрабатывается платёжной подсистемой.
- **Платёжная подсистема** связывает запросы на транзакцию, транзакции и электронные чеки.
- **Чек** хранит плательщика, идентификатор банка, через который была совершена транзакция, и количество потраченных средств.
- **Подсистема уведомлений** связывает транзакции, уведомления и пользователей системы.
- **Уведомление** хранит свой текст, и возможно связано с чеком - если уведомление было отправлено в результате транзакции.

- **Проездной билет** хранит своё время действия, количество оставшихся поездок, и тип проездного билета. Проездной билет должен находиться во владении одного и только одного пассажира.
- **Жалоба** хранит свой текст (описание неисправности в работе сайта, оставленное пользователем) и ссылку на нерабочий элемент приложения.
- **Маршрут** хранит номер маршрута, и связан с транспортом.
- **Расписание** хранит название остановки и время прибытия на данную остановку, и связано с транспортом.
- **Транспорт** хранит тип транспортного средства и количество посадочных мест. Транспорт может следовать по какому-то маршруту по определённому расписанию, поэтому он связан с соответствующими сущностями.
- **Местоположение транспорта** хранит координаты транспортного средства на карте мира (С.Д. / В.Ш.), а также текущую скорость транспортного средства.
- **Карта местности** хранит изображение карты со спутника, размеры изображения, и масштаб карты.

3. АРХИТЕКТУРНЫЕ РЕШЕНИЯ

3.1. Диаграмма Use Case.

На рисунке 12 представлена диаграмма Use Case.

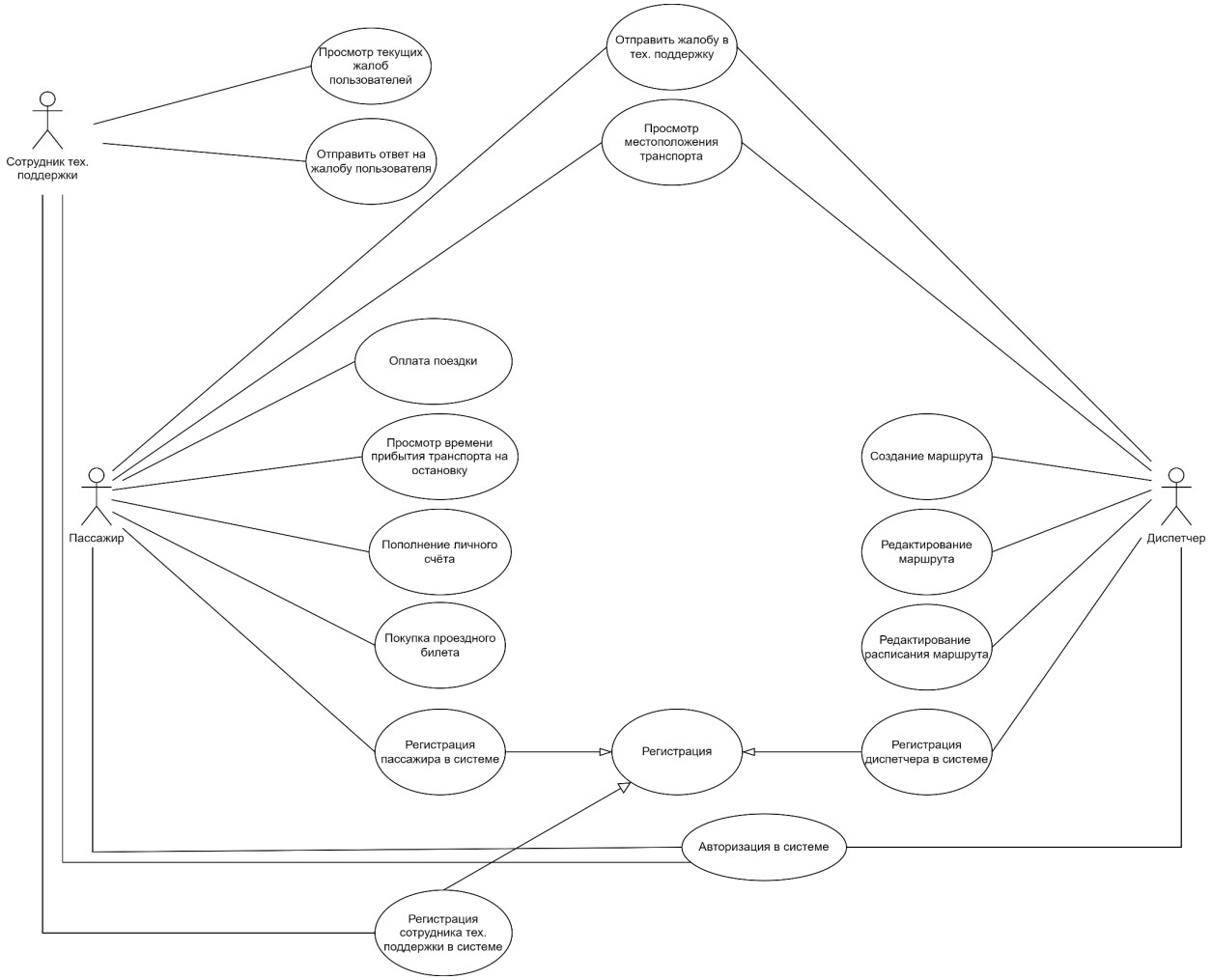

Рисунок 12. Диаграмма Use Case

- **Пассажир** имеет возможность авторизоваться и зарегистрироваться в системе, отправить жалобу в техническую поддержку, просмотреть местоположение транспорта на карте, посмотреть время прибытия транспорта на выбранную остановку, пополнить свой личный счёт, купить проездной билет или оплатить конкретную поездку.
- **Диспетчер** имеет возможность авторизоваться и зарегистрироваться в системе, отправить жалобу в техническую поддержку, просмотреть

местоположение транспорта на карте, создать или отредактировать маршрут, отредактировать расписание маршрута.

- **Сотрудник технической поддержки** имеет возможность авторизоваться и зарегистрироваться в системе, просмотреть текущие жалобы пользователей и отправлять ответы на них.

3.2. Диаграмма классов

На рисунке 13 представлена диаграмма классов.

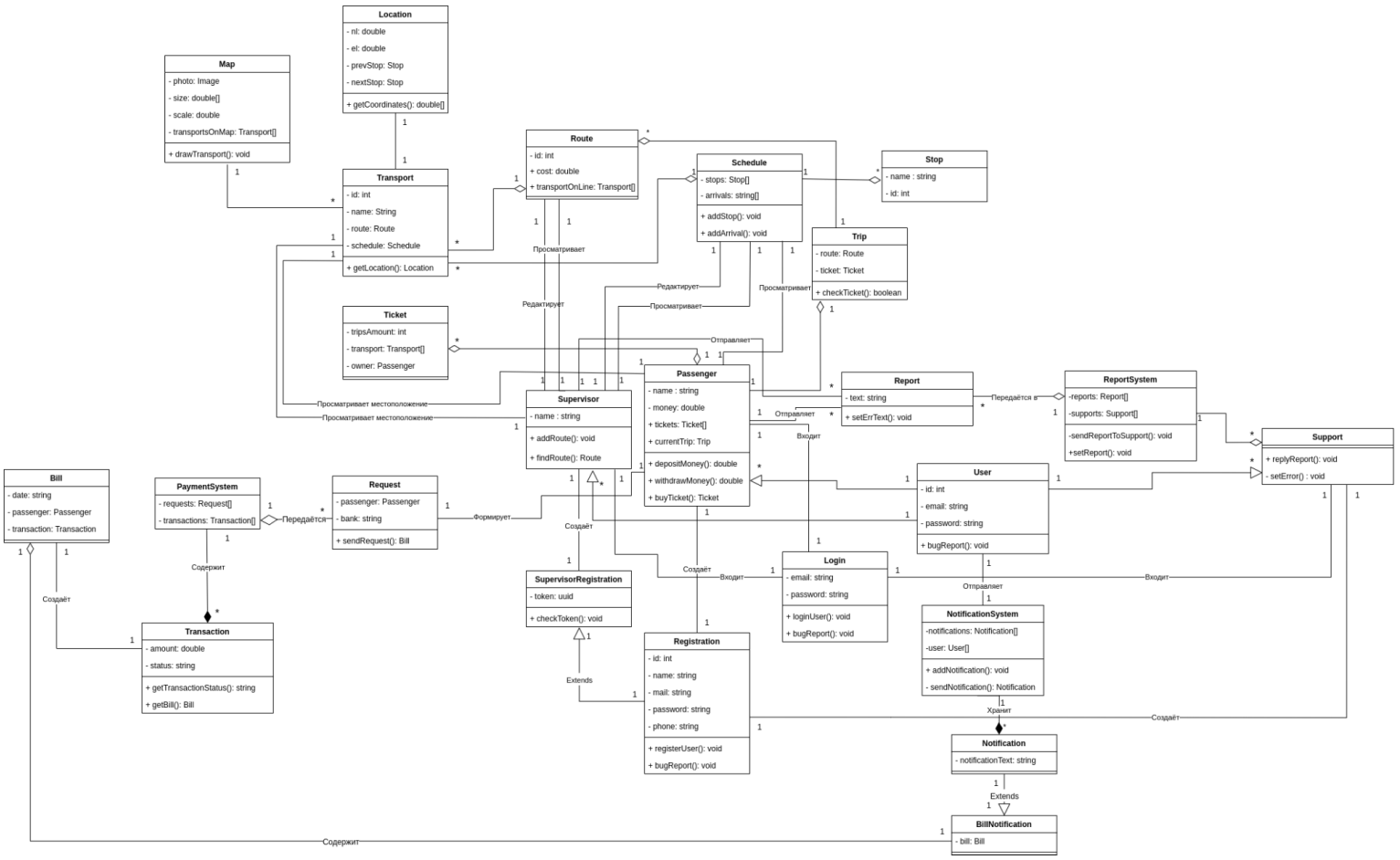

Рисунок 13. Диаграмма классов.

3.2.1. Описание классов

Класс **User** представляет собой сущность зарегистрированного в системе пользователя.

Поля:

- id - уникальный идентификатор пользователя в системе;
- email - адрес электронной почты пользователя;
- password - пароль пользователя в системе.

Методы:

- bugReport() - отправка отчёта об ошибке.

Класс **Passenger** представляет собой сущность зарегистрированного в системе пассажира.

Поля:

- name - ФИО пользователя;
- money - баланс личного счёта пользователя;
- tickets - проездные билеты, которыми владеет пользователь.
- currentTrip - текущая поездка пассажиры

Методы:

- depositMoney() - пополнение личного счёта;
- withdrawMoney() - вывод денег с личного счёта;
- buyTicket() - приобретение проездного билета.

Класс **Supervisor** представляет собой сущность зарегистрированного в системе диспетчера.

Поля:

- name - ФИО пользователя.

Методы:

- addRoute() - добавление маршрута.
- findRoute() - поиск маршрута

Класс **Support** представляет собой сущность зарегистрированного в системе сотрудника технической поддержки.

Методы:

- replyReport() - отправка ответа на жалобу пользователя;
- setError() - установка метки об ошибке в работе системы (для передачи жалобы другим сотрудникам технической поддержки или команде разработчиков).

Класс **Login** представляет собой вход пользователя в систему.

Поля:

- mail - адрес электронной почты пользователя;
- password - пароль от учётной записи в личном кабинете пользователя.

Методы:

- loginUser() - осуществление входа пользователя в систему;
- bugReport() - отправка отчёта об ошибке (от незарегистрированного в системе пользователя).

Класс **Registration** представляет собой регистрацию пользователя в системе.

Поля:

- id - идентификатор нового пользователя;
- name - ФИО нового пользователя;
- mail - адрес электронной почты нового пользователя;
- password - пароль от учётной записи нового пользователя;
- phone - номер мобильного телефона нового пользователя.

Методы:

- registerUser() - осуществление регистрации пользователя;
- bugReport() - отправка отчёта об ошибке (от незарегистрированного в системе пользователя).

Класс **SupervisorRegistration** представляет собой регистрацию диспетчера в системе.

Поля:

- token - выданный диспетчеру токен для регистрации.

Методы:

- checkToken() - проверка токена для регистрации в базе данных диспетчеров.

Класс **PaymentSystem** представляет собой систему оплаты.

Поля:

- requests - запросы на оплату;
- transactions - транзакции запросов на оплату.

Класс **Request** представляет собой запрос на оплату.

Поля:

- passenger - пользователь, сформировавший запрос.

- bank - банк, через который проводятся транзакции.

Методы:

- sendRequest() - отправка запроса на оплату.

Класс **Transaction** представляет собой транзакцию денежных средств.

Поля:

- amount - сумма денежных средств;
- status - текущее состояние транзакции (в процессе / успешно завершена / ошибка / недостаточно средств).

Методы:

- getTransactionStatus() - получение состояния транзакции;
- getBill() - создание чека.

Класс **Bill** представляет собой электронный чек.

Поля:

- date - дата выдачи чека;
- passenger - пассажир, которому был выдан чек;
- transaction - транзакция, информацию о которой хранит чек;

Класс **NotificationSystem** представляет собой систему оповещений.

Поля:

- notifications - оповещения, отправляемые пользователю;
- user - пользователь, которому приходят оповещения.

Методы:

- addNotification() - добавление оповещения в очередь;
- sendNotification() - отправка оповещения, находящегося в очереди.

Класс **Notification** представляет собой оповещение.

Поля:

- notificationText - текст оповещения.

Класс **BillNotification** представляет собой оповещение, содержащее электронный чек.

Поля:

- bill - электронный чек, прикреплённый к оповещению.

Класс **Ticket** представляет собой проездной билет.

Поля:

- tripsAmount - количество оставшихся поездок;
- transport - на какие транспортные средства распространяется действие проездного билета;
- owner - владелец проездного билета;

Класс **Transport** представляет собой транспортное средство.

Поля:

- id - уникальный идентификатор транспортного средства;
- name - название транспортного средства;
- route - текущий маршрут следования транспортного средства.
- shedule - расписание, по которому следует текущее ТС.

Методы:

- getLocation() - получение текущего местоположения транспортного средства.

Класс **Location** представляет собой местоположение транспортного средства.

Поля:

- nl - координаты ТС на карте мира (северная широта);
- el - координаты ТС на карте мира (восточная долгота);
- prevStop - предыдущая остановка на маршруте;
- nextStop - следующая остановка на маршруте;

Методы:

- getCoordinates() - получение текущих координат ТС на карте мира (С.Ш. и В.Д.).

Класс **Map** представляет собой карту, отображающую местоположение транспортных средств.

Поля:

- photo - изображение карты, на которой будут показаны ТС.
- size - размеры карты.
- scale - текущий масштаб карты.
- transportOnMap - массив ТС, отображающихся на карте.

Методы:

- drawTransport() - отрисовка ТС на карте.

Класс **Route** представляет собой маршрут следования ТС.

Поля:

- id - уникальный идентификатор маршрута;
- cost - стоимость проезда по маршруту;
- transportOnLine - транспорт, следующий по маршруту.

Класс **Schedule** представляет собой расписание ТС.

Поля:

- stops - остановки в расписании;
- arrivals - время прибытия на остановки в расписании.

Методы:

- addStop() - добавление остановки в расписание;
- addArrival() - добавление времени прибытия в расписание;

Класс **Stop** представляет собой остановку на маршруте.

Поля:

- name - название остановки;
- id - уникальный идентификатор остановки.

Класс **Report** представляет собой отчёт об ошибке.

Поля:

- text - содержание оповещения.

Методы:

- setErrText() - задание содержания оповещения.

Класс **ReportSystem** представляет собой систему отправки отчётов об ошибках.

Поля:

- reports - неразрешённые отчёты об ошибках;
- supports - сотрудники службы технической поддержки.

Методы:

- sendReportToSupport() - отправка отчёта об ошибке сотруднику технической поддержки.
- setReport() - добавление отчёта об ошибке в список неразрешённых.

Класс **Trip** представляет собой текущую поездку пассажира.

Поля:

- route - маршрут следования.
- ticket - по какому билету будет оплачен проезд

Методы:

- checkTicket() - проверка оплаты проезда билетом.

3.2.2 Описание процессов

3.2.2.1 Авторизация пассажира

При открытии веб-приложения, пассажир переадресовывается на страницу с авторизацией где, если предыдущая сессия не истекла по времени, происходит автоматическая авторизация при помощи класса Login, иначе, пользователю предложат ввести свои данные для входа(email, пароль). Если пользователя с данным email не существует в базе данных, ему предложат зарегистрироваться в приложении и перенаправят на страницу с регистрацией, реализованной с помощью класса Registration и метода registerUser(). Таким образом, вошедший пользователь идентифицируется объектом класса Passenger, который наследуется от класса User.

3.2.2.2 Авторизация диспетчера

Диспетчер заходит в приложение при помощи выделенного для диспетчеров URL, попадая на страницу авторизации. Если диспетчер уже был зарегистрирован, он входит при помощи своего email и пароля, иначе ему предлагается регистрация, реализованная при помощи класса SupervisorRegistration, наследуемого от класса Registration. Также диспетчеру выдается специальный токен, при помощи которого он может быть зарегистрирован в приложении. Таким образом, сторонний пользователь не может быть зарегистрирован как диспетчер. Вошедший диспетчер идентифицируется объектом класса Supervisor, который наследуется от класса User.

3.2.2.3 Просмотр местоположения транспорта

С главной страницы пассажира он может перейти на страницу с отображением местоположения транспорта, с возможностью просмотра каждого ТС на карте, берущей информацию из класса Transport при помощи метода getLocation(), возвращающего объект

класса Location, хранящий северную широту(nl) и восточную долготу(el), а также предыдущую и следующую остановки(prevStop, nextStop).

3.2.2.4 Просмотр времени прибытия транспорта на остановку

С главной страницы пассажира он может перейти на страницу с расписанием с возможностью выбора интересующей его остановки и маршрута. Обращаясь к полям класса Schedule - stops и arrivals приложение получает остановку и время прибытия текущего маршрута на ней. Само расписание в виде объекта вышеупомянутого класса хранится в поле класса Route.

3.2.2.5 Пополнение пассажиром личного счёта

Пассажир после авторизации может перейти в окно пополнения своего личного счёта. После заполнения необходимой информации функцией depositMoney() отправляется запрос (request) в PaymentSystem. Система оплаты создает транзакцию (Transaction), содержащую сумму и статус транзакции. При проведении успешной транзакции между клиентом и банком, класс создаёт чек (Bill) и возвращает его пользователю функцией getBill().

3.2.2.6 Покупка проездного билета

После пополнения счета, пользователь может купить себе проездные билеты. Это осуществляется на отдельной странице приложения. Транзакция между банком не нужна, так как пользователь использует уже переведённые средства. С помощью метода buyTicket() пассажир получает билет (Ticket), который сохраняется в массив билетов (tickets). Билет содержит количество оставшихся поездок, доступные транспортные средства (Transport) и владельца проездного билета (Passenger), для повторной проверки при использовании.

3.2.2.7 Отправка жалобы в тех. поддержку

В случае возникновения ошибки перед пользователем появится окно выбора в котором он может отправить сообщение об ошибке (Report). Ошибка поступает в ReportSystem, где, либо передаётся в тех. поддержку методом sendReportToSupport, либо сохраняется как нерешенный, если сотрудник тех. поддержки не способен его решить.

3.2.2.8 Создание маршрута

С главной страницы диспетчера он может войти на вкладку создания маршрута. При помощи метода addRoute() у диспетчера есть возможность создать новый маршрут. Диспетчер может взаимодействовать с расписанием при помощи класса Schedule, в котором присутствуют методы для добавления остановки (addStop()), а также времени прибытия транспорта на остановку. Данный класс хранит массив остановок stops типа Stop[], а также массив времени прибытия транспорта arrivals типа string[]. Также у диспетчера есть возможность взаимодействия с массивом маршрутов Route, у которого присутствует переменная schedule тима Schedule, отвечающая за движение транспорта по маршруту, массив транспорта, находящегося на линии, transportOnLine типа Transport[]. Также класс Route имеет такие поля, как id типа int, который обозначает номер маршрута, а также стоимость билета на данный маршрут cost типа double.

3.2.2.9 Редактирование маршрута

Диспетчер имеет возможность редактировать маршруты движения транспорта. Он может взаимодействовать с классом Route, изменяя в нем определенные поля: cost, schedule, transportOnLine при помощи формы, отображающейся на странице.

3.2.2.9 Редактирование расписания

Диспетчер имеет возможность редактировать расписание движения транспорта. Он имеет доступ к полям класса Schedule, тем самым может изменять как остановки, хранящиеся в соответствующем массиве, так и время прибытия транспорта на необходимую остановку, хранящееся в массиве arrivals.

3.3. Диаграммы объектов

На рисунке 14 изображена диаграмма объектов, описывающая процесс покупки проездного билета пассажиром.

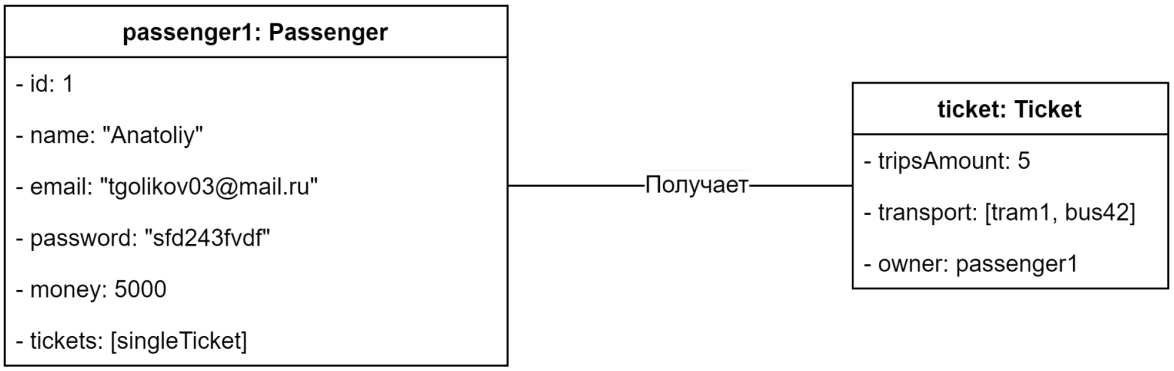

Рис. 14. Покупка билета

Объект passenger1 связан композицией с объектом ticket, так как пассажир получает билет при его покупке, и все приобретённые билеты хранятся в объекте passenger1.

На рисунке 15 изображена диаграмма объектов, описывающая процесс пополнения счёта пассажиром для дальнейшей покупки билета.

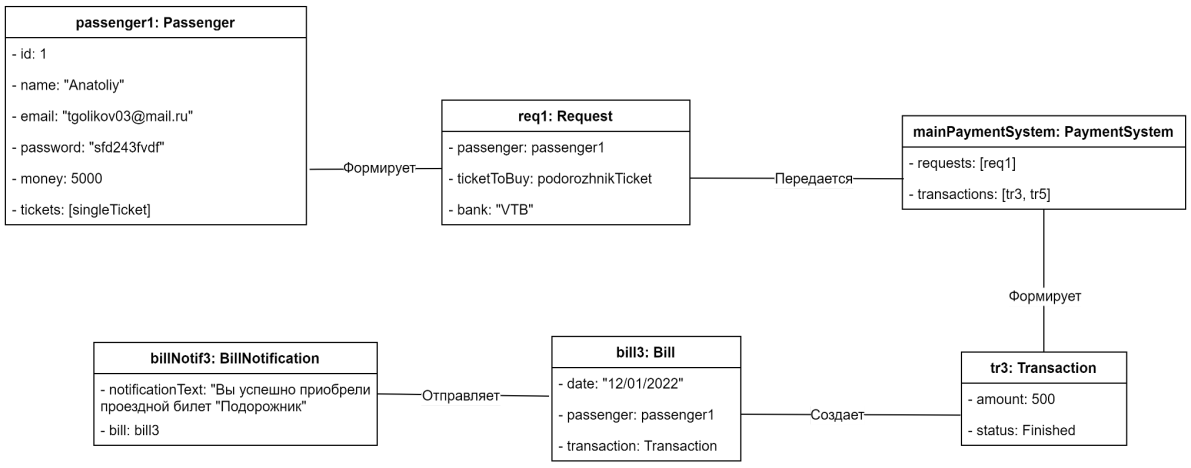

Рис. 15. Пополнение счёта

Объект passenger1 связан композицией с объектом req1, так как пассажир формирует запрос, который хранит ссылку на пассажира.

Объект req1 связан с объектом mainPaymentSystem композицией, так как запрос передается в платежную систему.

Объект mainPaymentSystem содержит объект tr3, так как платежная система формирует ответ об успехе транзакции и хранит результат операции.

Объект tr3 связан композицией с объектом bill3, так как результат транзакции создаёт чек.

Объект billNotif3 содержит объект bill3, так как чек отправляется в систему уведомлений об оплате и становится частью этой системы.

На рисунке 16 изображена диаграмма объектов, описывающая процесс просмотра пассажиром текущего местоположения транспортного средства на маршруте.

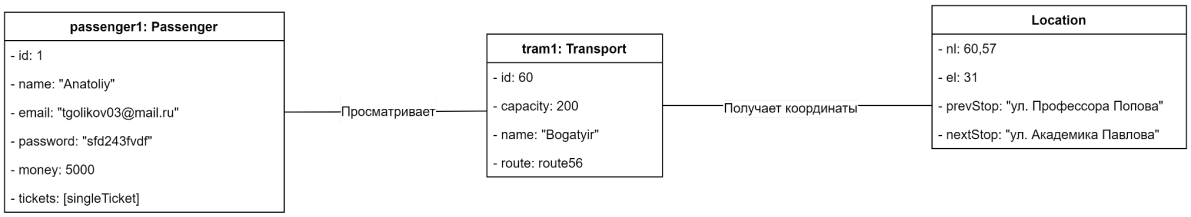

Рис. 16. Просмотр местоположения транспорта

Объект passenger1 связан с объектом tram1 ассоциацией, так как пассажир может просматривать местоположение выбранного транспортного средства.

Объект tram1 связан с объектом Location ассоциацией, так как для обеспечения пассажиру возможности отслеживать местоположение ТС необходимо, чтобы ТС получило свои текущие координаты и затем отобразило их пользователю.

На рисунке 17 изображена диаграмма объектов, описывающая процесс просмотра пассажиром времени прибытия ТС на остановку.

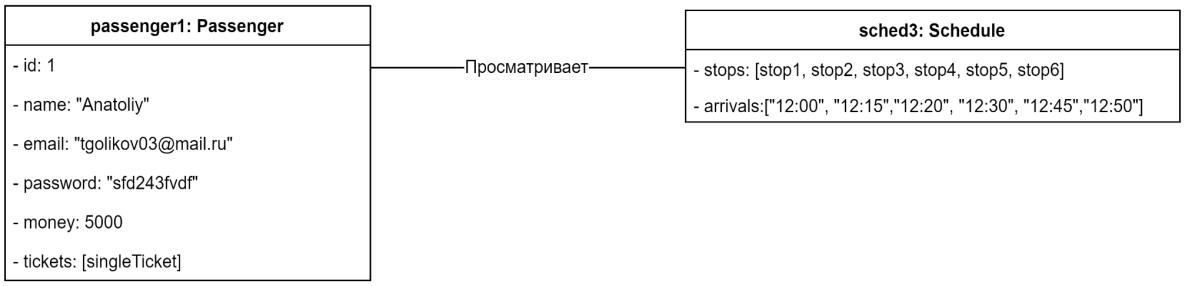

Рис. 17. Просмотр времени прибытия ТС на остановку

Объект passenger1 связан ассоциацией с объектом sched3, так как пассажир имеет возможность просматривать на какую остановку и когда прибудет выбранное транспортное средство.

На рисунке 18 изображена диаграмма объектов, описывающая процесс создания маршрута в рамках ИС СКТ.

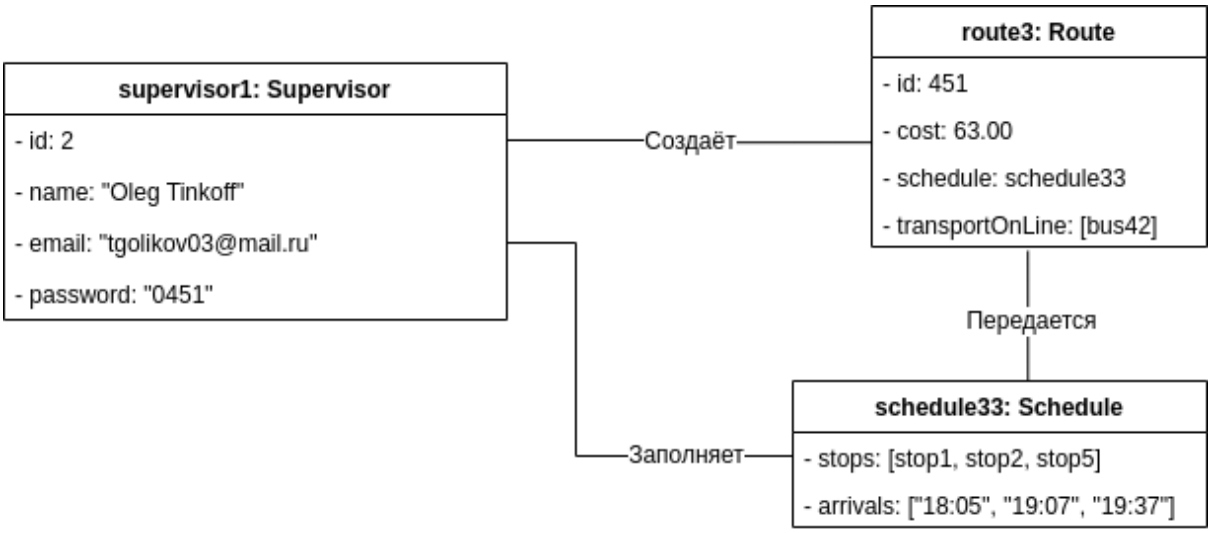

Рис. 18. Создание маршрута.

Объект supervisor1 связан с объектом route3 ассоциацией, так как пользователь (диспетчер) создаёт новый маршрут.

Объект supervisor1 связан с объектом schedule33 ассоциацией, так как пользователь (диспетчер) заполняет список остановок и время прибытия для расписания созданного маршрута.

Объект route3 содержит объект schedule33, так как каждый маршрут должен содержать список остановок, которые отдельно задаются диспетчером.

На рисунке 19 изображена диаграмма объектов, описывающая внесение изменений в маршрут.

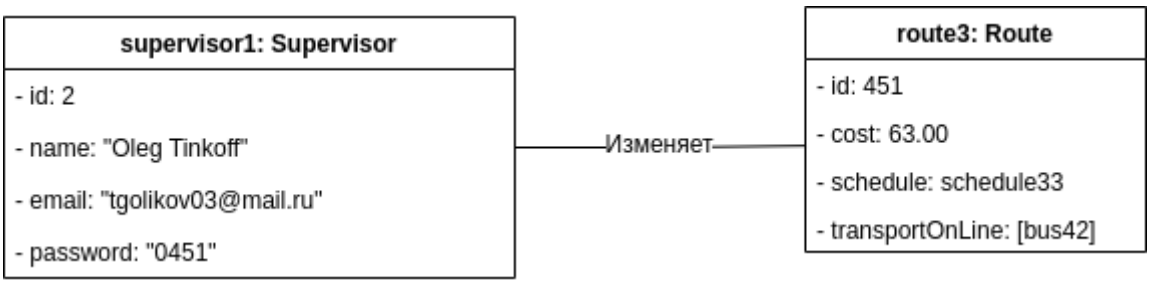

Рис. 19. Редактирование маршрута

Объект supervisor1 связан ассоциацией с объектом route3, так как диспетчер изменяет уже имеющийся в системе маршрут.

На рисунке 20 изображена диаграмма объектов, описывающая процесс внесения изменений в расписание на конкретном маршруте.

Рис. 20. Редактирование расписания.

Объект supervisor1 связан с объектом schedule33 ассоциацией, так как диспетчер редактирует количество остановок, а также, в которое на них должно прибыть ТС на конкретном маршруте.

Объект route3 содержит объект schedule33, так как каждый маршрут хранит список актуальных остановок и актуальный список интервалов прибытия на каждую остановку, соответственно, после изменений расписания в списке остановок нужно отразить это изменение и в данных маршрута.

На рисунке 21 изображена диаграмма объектов, описывающая отправку жалобы в техническую поддержку.

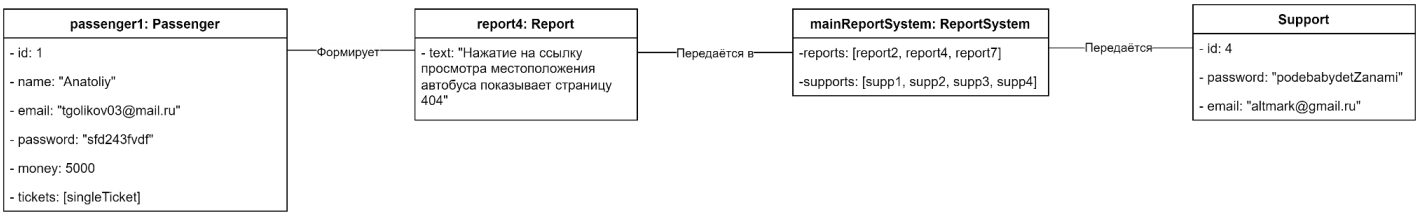

Рис. 21. Отправка жалобы в тех. поддержку

Объект passenger1 связан с объектом report4 ассоциацией, так как пользователь формирует сообщения о полученной ошибке, которые затем будут переданы сотруднику тех. поддержки.

Объект mainReportSystem содержит report4, поскольку все технические проблемы решаются единой группой сотрудников тех. поддержки.

Объект mainReportSystem связан композицией с объектом supp4, так как сотрудники тех. поддержки являются частью системы технической поддержки, а также они получают сведения о проблемах, полученных от пользователей.

3.4. Диаграммы состояний

На рисунке 22 представлена диаграмма состояния маршрута.

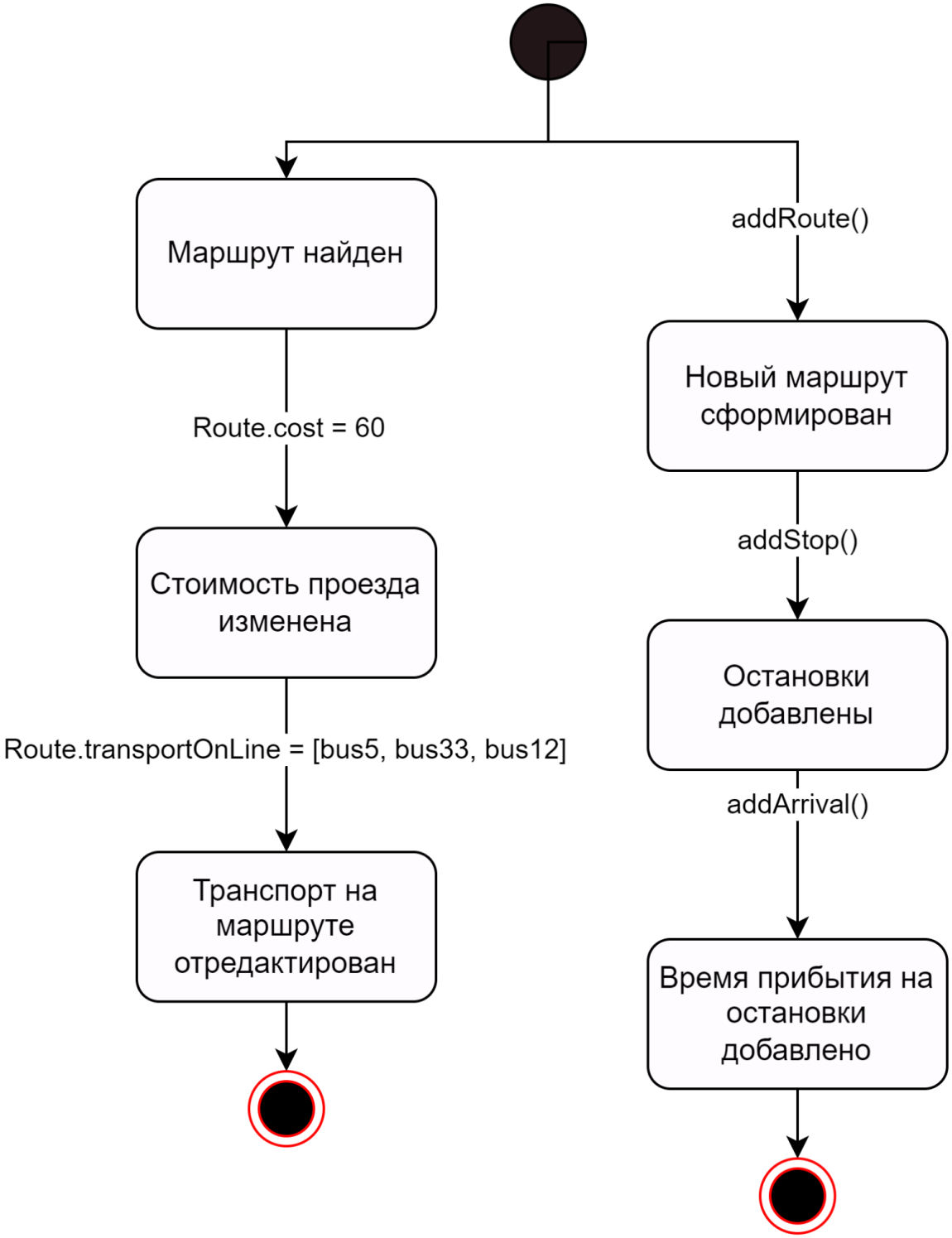

Рисунок 22. Диаграмма состояния маршрута.

В начальный момент времени маршрут может иметь состояние “маршрут найден”, когда диспетчер хочет отредактировать желаемый маршрут, иначе, если ему необходимо создать новый, маршрут имеет начальное состояние маршрут сформирован. Далее, при редактировании маршрута он переходит в состояния “Стоимость проезда изменена”, “Транспорт на маршруте отредактирован” при изменении параметров маршрута диспетчером. При создании нового маршрута, он переходит из состояния “Новый маршрут сформирован” в “Остановки добавлены”, “Время прибытия добавлено” при добавлении остановок на маршрут и редактировании его расписания. При создании маршрута транспорт на нем инициализируется пустым массивом, в который затем добавляется транспорт при помощи редактирования маршрута.

На рисунке 23 представлена диаграмма состояния личного счета пассажира.

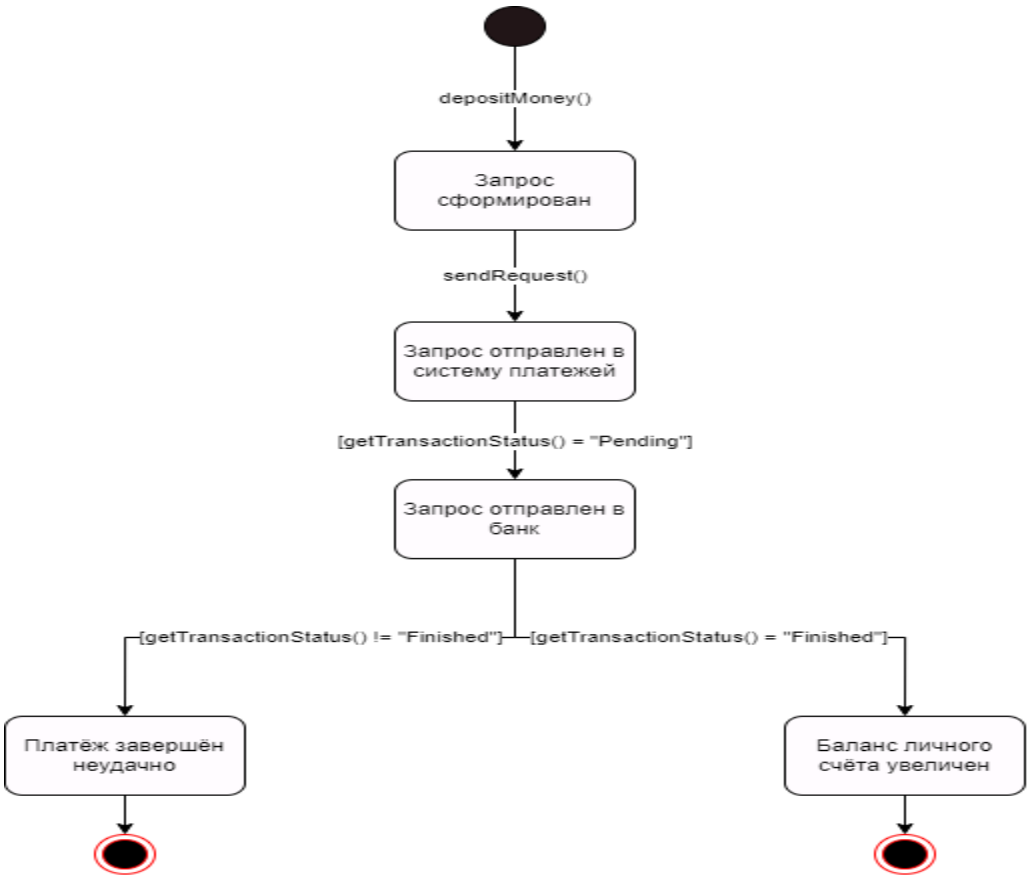

Рисунок 23. диаграмма состояния личного счета пассажира

При необходимости пополнить баланс, личный счет пассажира имеет статус “Запрос сформирован”. После успешного формирования запроса, он отправляется в систему платежей и получает статус “Запрос отправлен в систему платежей”, затем система формируется транзакцию, содержащую запрос, и отправляет её в банк - статус “Запрос отправлен в банк”. Если транзакция была выполнена успешно, личный счет приобретает статус “Баланс личного счета увеличился”, иначе - “Платеж завершен неудачно”.

На рисунке 24 представлена диаграмма состояния проездного билета.

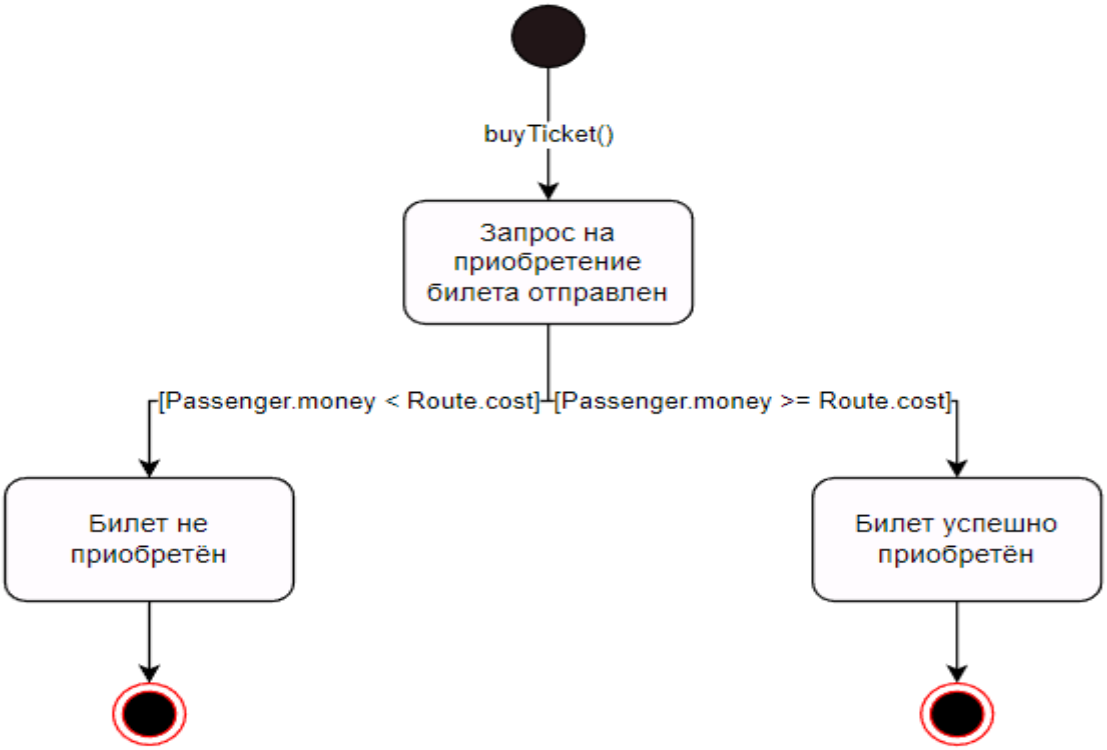

Рисунок 24. Диаграмма состояния проездного билета.

На рисунке 24 представлена диаграмма состояния проездного билета. В начальный момент времени покупка билета находится в состоянии “Запрос на приобретение билета отправлен”. После получения запроса на приобретение возможны два варианта. Если количество средств на счёте пассажира меньше стоимости билета на маршрут, то состояние операции устанавливается в “Билет не приобретен” с дальнейшим

переходом в заключительное состояние. В противоположном случае, если количество средств пассажира больше или равно стоимости билета на маршрут, то состояние устанавливается на “Билет успешно приобретен”, после чего также осуществляется переход в заключительное состояние.

На рисунке 25 представлена диаграмма состояния отчёта об ошибке.

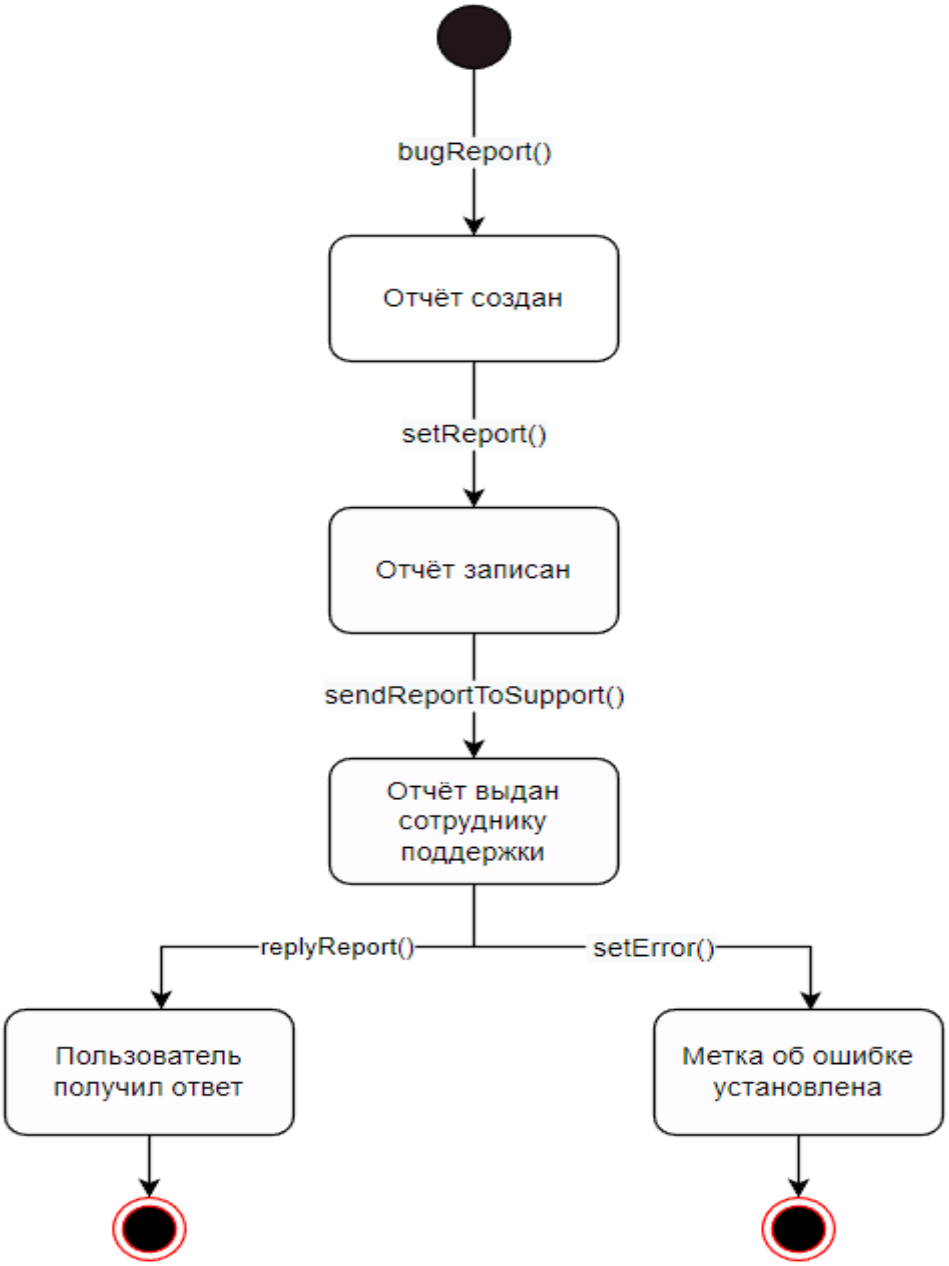

Рисунок 25. Диаграмма состояния отчета об ошибке.

На рисунке 25 представлена диаграмма состояния отчёта об ошибке. В начальный момент времени, после отправки жалобы пользователем,

отчет об ошибке имеет статус “Отчёт создан”. Когда отчет об ошибке будет создан, он будет помещена в хранилище отчетов об ошибках, а также иметь статус “Отчёт записан”. После того, как сотрудник технической поддержки выберет соответствующую жалобу, данной жалобе будет установлен статус “Отчёт выдан сотруднику поддержки”. Затем, после того, как сотрудник технической поддержки получает отчет об ошибке пользователя, отчету присваивается один из статусов: “Метка об ошибке установлена” – в случае, если сотрудник технической поддержки не может решить проблему, после чего, перевод в заключительное состояние, или “Пользователь получил ответ” – в случае, если проблема была решена, после чего, также перевод в заключительное состояние.

На рисунке 26 представлена диаграмма состояния оповещения пользователей.

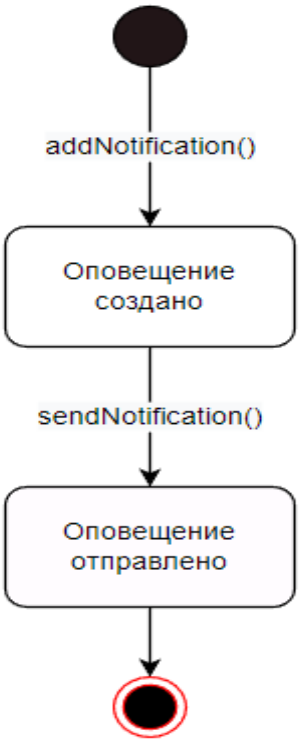

Рисунок 26. Диаграмма состояния оповещения пользователя.

В начальный момент времени оповещение имеет статус “Оповещение создано”. После отправки оповещения пользователю

устанавливается статус “Оповещение отправлено”, после чего осуществляется переход в заключительное состояние.

3.5. Диаграмма деятельности

На рисунке 27 представлена диаграмма деятельности.

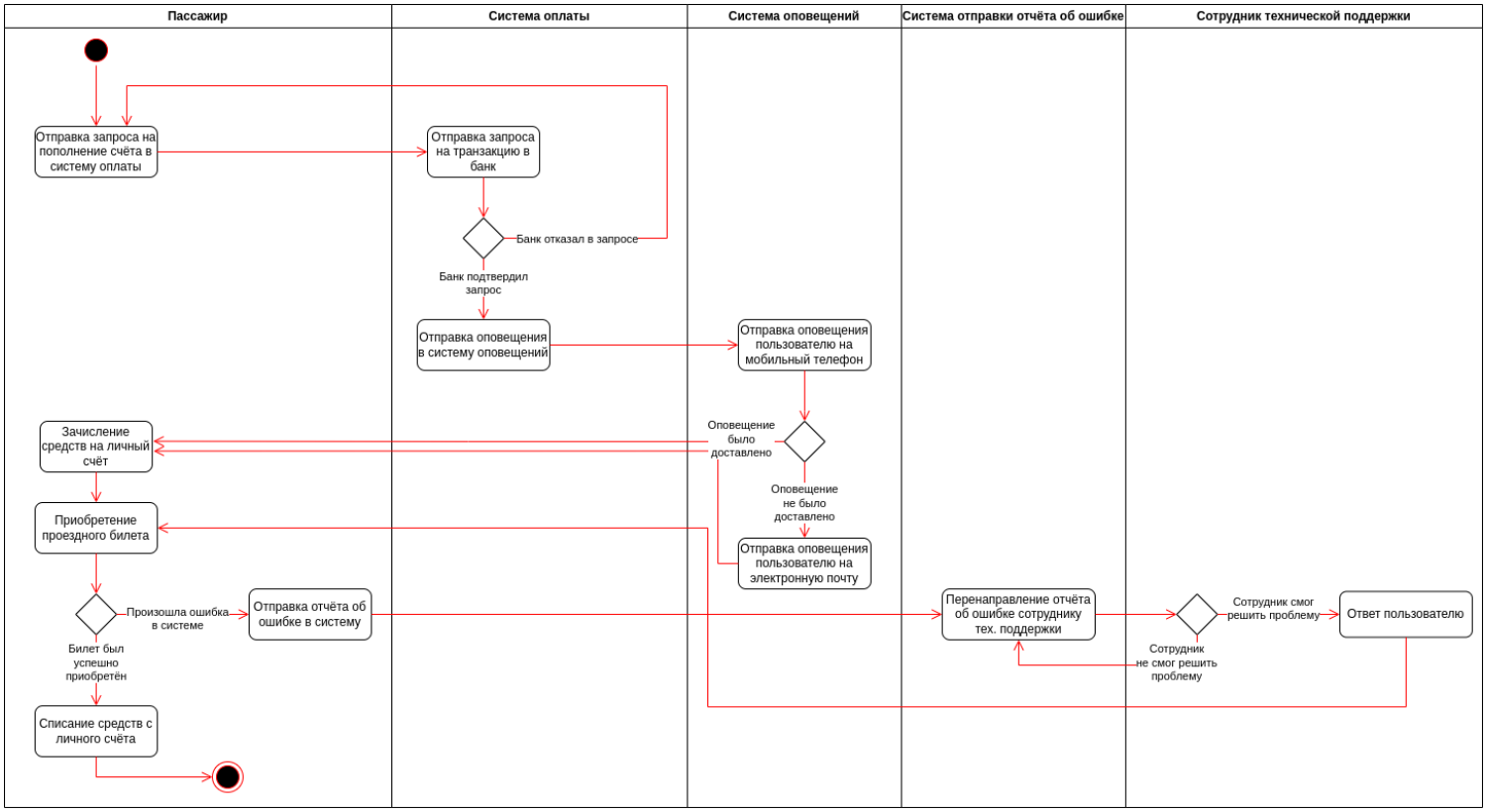

Рис. 27. Диаграмма деятельности

Данная диаграмма деятельности иллюстрирует переход управления в частях подсистем при пополнении пользователем своего личного счёта и приобретении проездного билета.

Изначально пользователь (пассажир) отправляет запрос на пополнение счёта в систему оплаты, которая, в свою очередь, отправляет запрос на осуществление транзакции в банк. Если банк отказывает в проведении транзакции, пользователю предлагается повторная попытка оплаты. Если банк подтверждает успешное завершение транзакции, то затем создаётся оповещение и отправляется в систему оповещений.

Система оповещений сначала пытается отправить оповещение пользователю на мобильный телефон, а если это не удалось, то на

электронную почту. Независимо от того, дошло ли до пользователя уведомление по электронной почте, пассажиру предлагается на зачисленные средства приобрести проездной билет.

Если во время приобретения проездного билета в системе произошла ошибка, то пользователю предлагается автоматически сформировать отчёт и отправить его в систему отправки отчёта об ошибке. Данная система перенаправляет отчёт об ошибке любому из свободных сотрудников технической поддержки. Если сотрудник смог решить проблему, то он отправляет ответ пользователю о том, что проблема устранена, иначе ошибка передаётся другому сотруднику технической поддержки. После окончательного устранения проблемы пользователю предлагается повторно приобрести проездной билет.

После успешного приобретения проездного билета с личного счёта пользователя списываются средства, необходимые для приобретения проездного билета, после чего осуществляется переход в заключительное состояние.

3.6 Диаграмма последовательности

На рисунке 28 изображена диаграмма последовательности для отправки отчёта об ошибке. На диаграмме показано взаимодействие пользователя (:User) с системой отправки и обработки ошибок. Методом sendReport пользователь создаёт ошибку, после чего метод setReport отправляет её в систему (:ReportSystem). Система обработки ошибок распределяет полученные сообщения по доступным работникам поддержки (:Support). Сотрудник поддержки обрабатывает полученную ошибку и либо помогает решить проблему, возвращая Reply, либо сохраняет ошибку со специальной меткой в :ReportSystem.

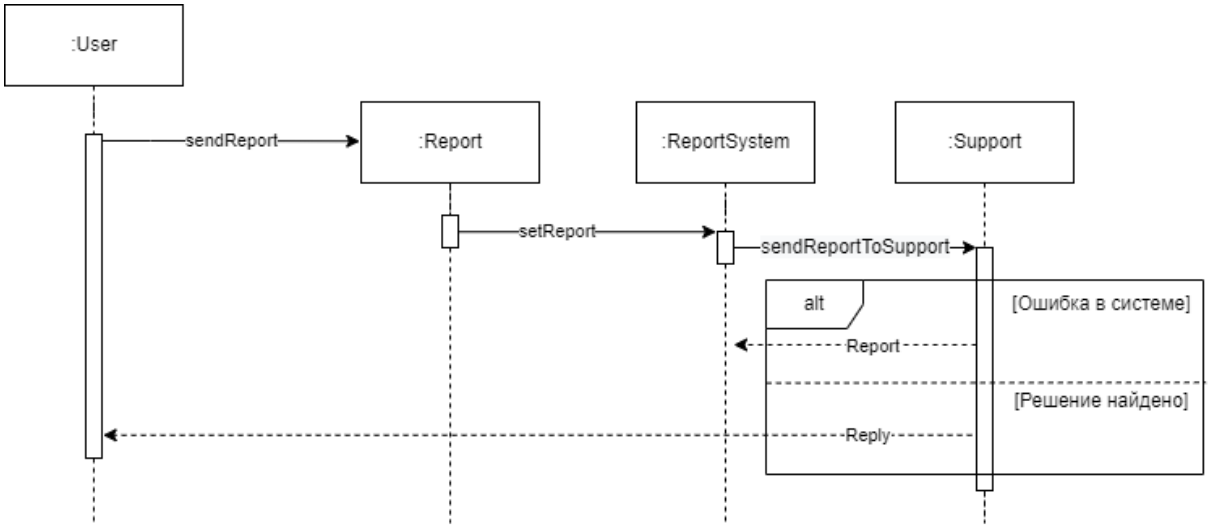

Рисунок 28. Диаграмма последовательности отправки сообщения об ошибке

На рисунке 29 показана диаграмма последовательности просмотра местоположения транспортного средства на маршруте. :User с помощью метода getLocation() обращается к :Transport, который в свою очередь методом getCoordinates() получает своё текущее местоположение у класса :Location и возвращает его в класс :Transport, который в свою очередь передаёт местоположение всех транспортных средств в класс :Map,

который отрисовывает карту пользователю (возвращает полученную карту (Map) пользователю (:User)).

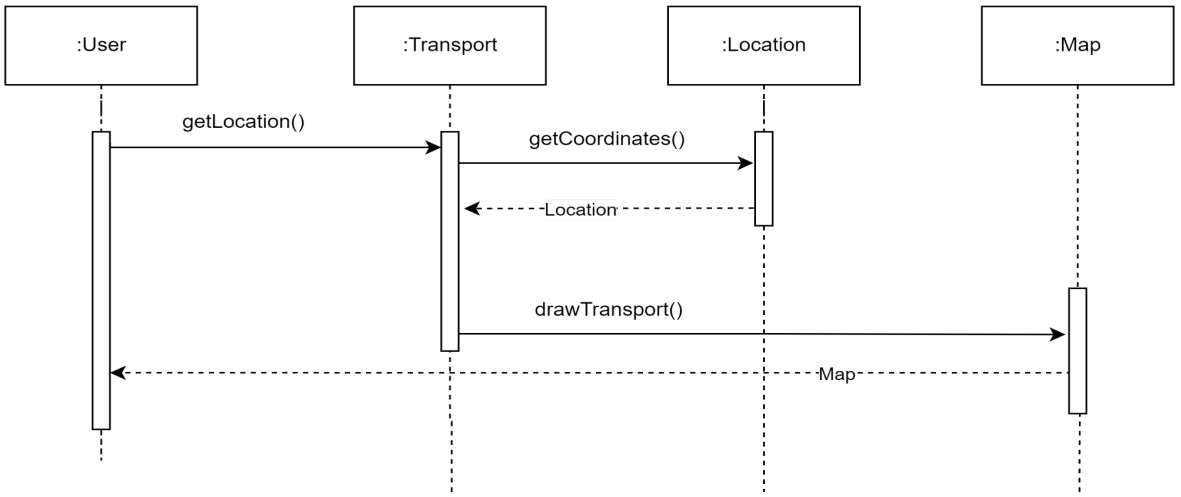

Рисунок 29. Диаграмма последовательности просмотра местоположения маршрута

На рисунке 30 представлена диаграмма пополнения счёта пассажира в ИС СКТ. :Passenger с помощью метода depositMoney() обращается к :Request, который в свою очередь методом sendRequests() формирует и отправляет запрос классу :PaymentSystem. Данный класс в свою очередь с помощью метода addNotification() формирует ответ об успехе транзакции и добавляет его в систему уведомлений (класс :NotificationSystem). Этот класс возвращает сформированный в результате чек (Bill) пассажиру (:Passenger).

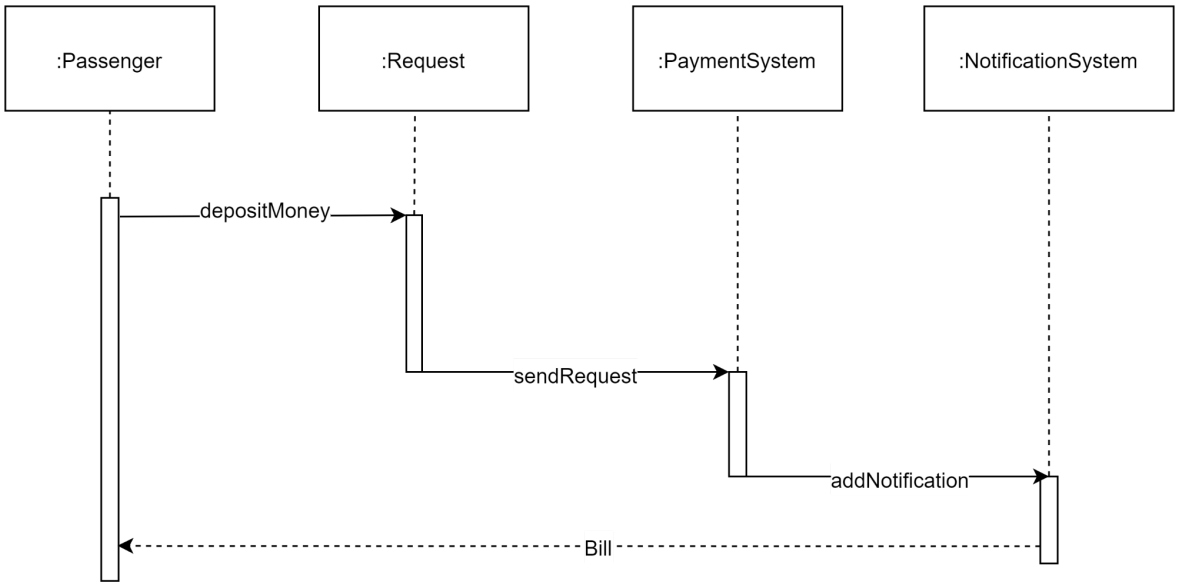

Рисунок 30. Диаграмма последовательности пополнения счёта

На рисунке 31 представлена диаграмма редактирования/создания маршрута в ИС СКТ. У данной последовательности действий существует два варианта развития. В случае создания нового маршрута класс :Supervisor добавляет новую запись в список всех маршрутов (класс :Route), хранящихся в ИС, при помощи метода addRoute(), а затем для нового маршрута класса :Route формируется расписание при помощи методов addStop() (добавляет остановку) и addArrival() (добавляет время прибытия ТС на остановку). В результате класс :Route возвращает сформированный маршрут Route диспетчеру.

В случае редактирования имеющегося маршрута класс :Supervisor обращается к существующему маршруту при помощи метода findRoute(), а в классе :Route, на линию выводится новое транспортное средство, введённое диспетчером, заменяя уже имеющееся, и изменяется стоимость проезда по маршруту. После завершения редактирования диспетчеру (класс Supervisor) возвращается обновленный маршрут Route.

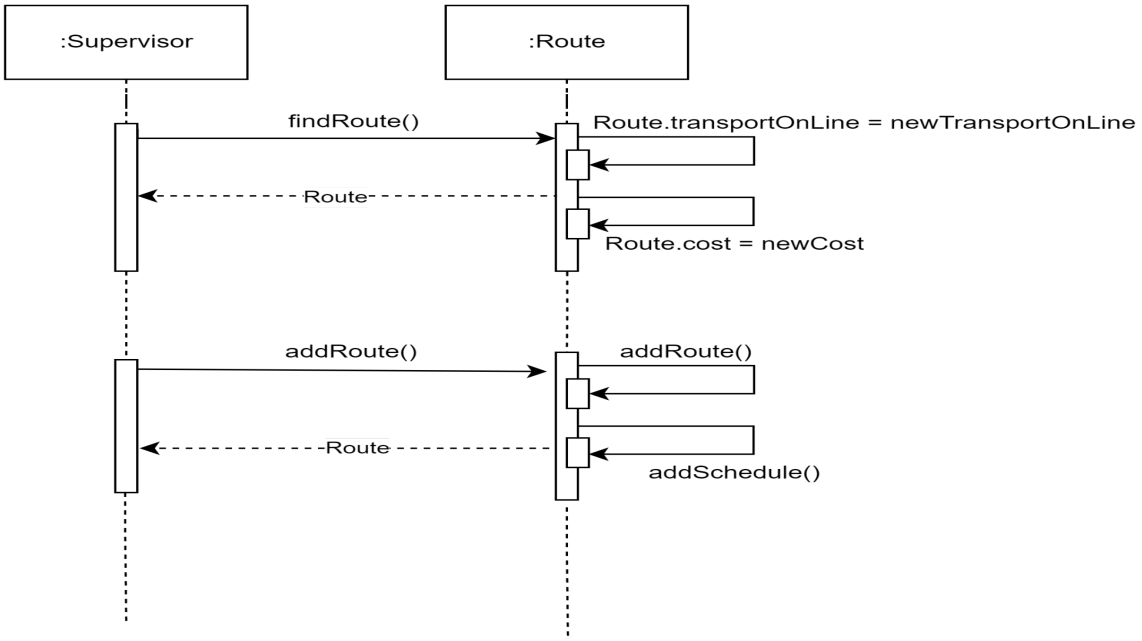

Рисунок 31. Диаграмма последовательности создания/редактирования маршрута

3.7 Диаграмма коммуникаций

На рисунке 32 представлена диаграмма коммуникаций пополнения счёта.

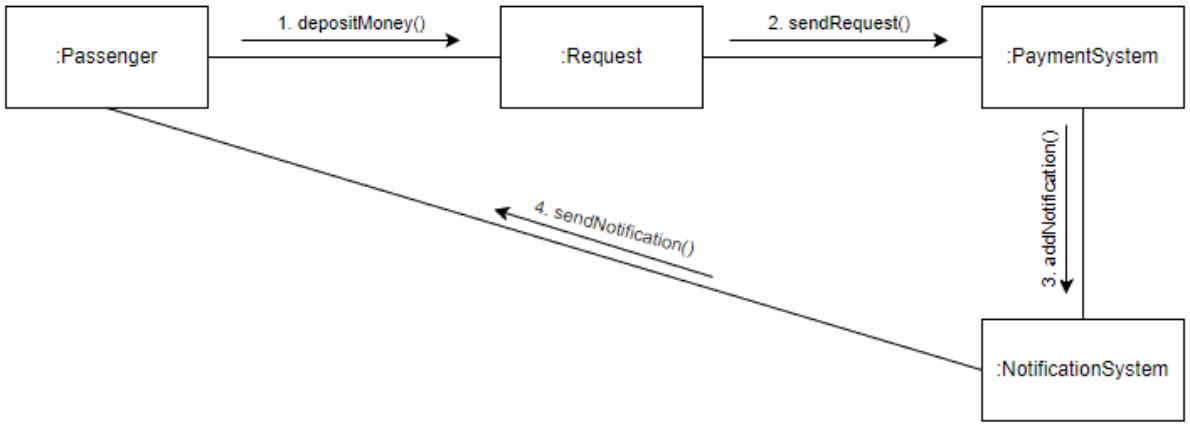

Рисунок 32 - Диаграмма коммуникаций пополнения счёта.

На данной диаграмме представлена последовательность действий для пополнения счёта пользователя. Изначально, в случае, если пассажир хочет пополнить свой счет, то создается запрос (экземпляр класса Request) при помощи метода depositMoney(). Далее, данный запрос передается

платежной системе (экземпляру класса PaymentSystem) при помощи метода sendRequest(). После проведенной транзакции платежная система создает уведомление о проведенной транзакции (экземпляр класса NotificationSystem) при помощи метода addNotification() , а далее происходит отправка данного уведомления лицу, совершающему платеж (экземпляру класса Passenger) путем вызова метода sendNotification().

На рисунке 33 показана диаграмма коммуникации для процесса отправки отчёта об ошибке.

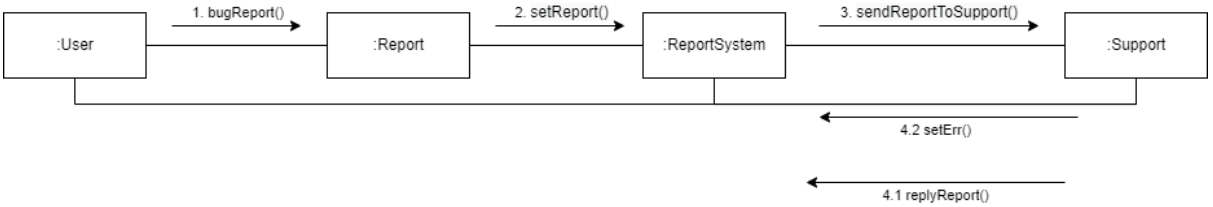

Рисунок 33. Диаграмма коммуникаций для отправки отчёта об ошибке

Диаграмма отражает процесс отправки отчёта об ошибке, которая возникает в процессе работы с программой. Методом bugReport() экземпляр класса Report передаётся в систему распределения отчётов. После чего ReportSystem выдаёт отчёт об ошибке экземпляру класса Support. Сотрудник поддержки, в свою очередь, может либо отослать ответ пользователю, либо сохранить в системе с меткой ошибки.

На рисунке 34 изображена диаграмма коммуникации для создания маршрута.

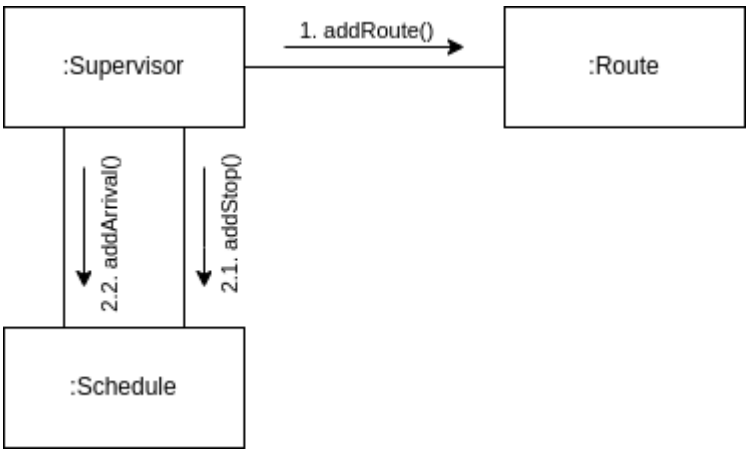

Рисунок 34. Диаграмма коммуникаций для создания маршрута

Создание маршрута реализовано методом addRoute() для экземпляра Supervisor. С помощью интерфейса сотрудник депо добавляет остановки (addStop()) и время прибытия на них (addArrival()), составляя расписание (Schedule) маршрута.

На рисунке 35 изображена диаграмма для редактирования маршрута.

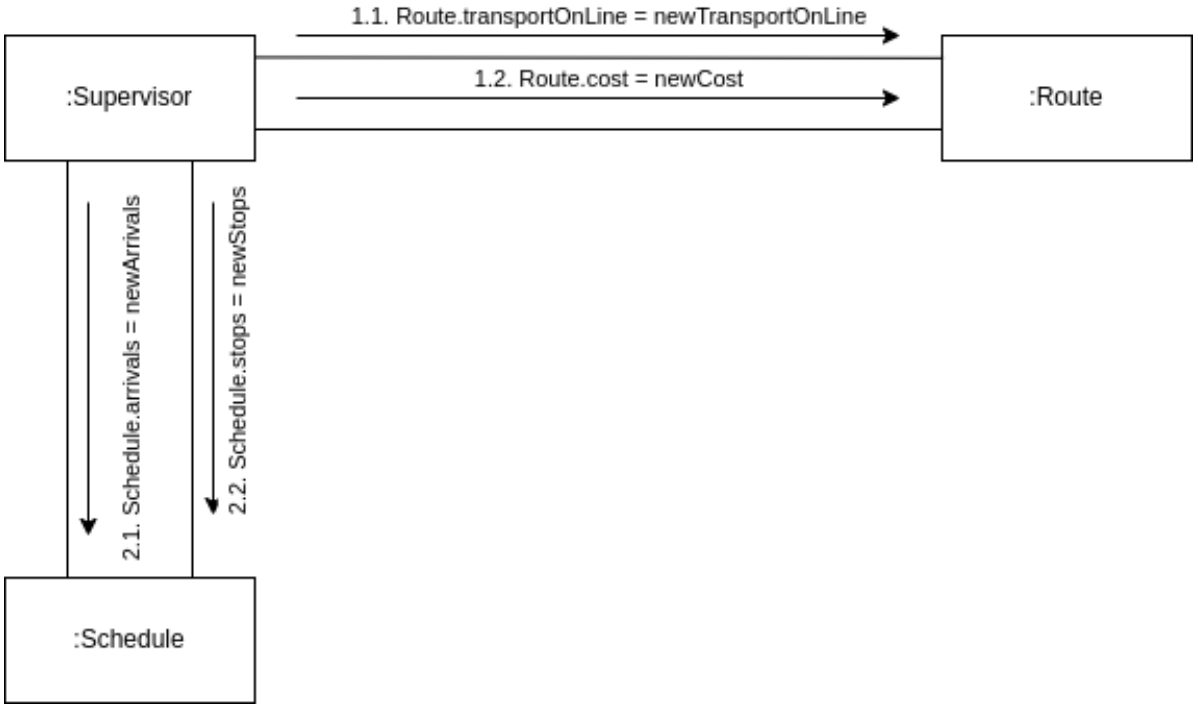

Рисунок 35. Диаграмма коммуникаций для редактирования маршрута

Редактирование маршрута реализовано схожим образом с созданием маршрута. Сначала происходит редактирование параметров маршрута: изменяются поля transportOnLine и cost. Затем происходит редактирование расписания: изменяются поля arrivals и stops.

На рисунке 36 изображена диаграмма коммуникации для просмотра местоположения транспортного средства.

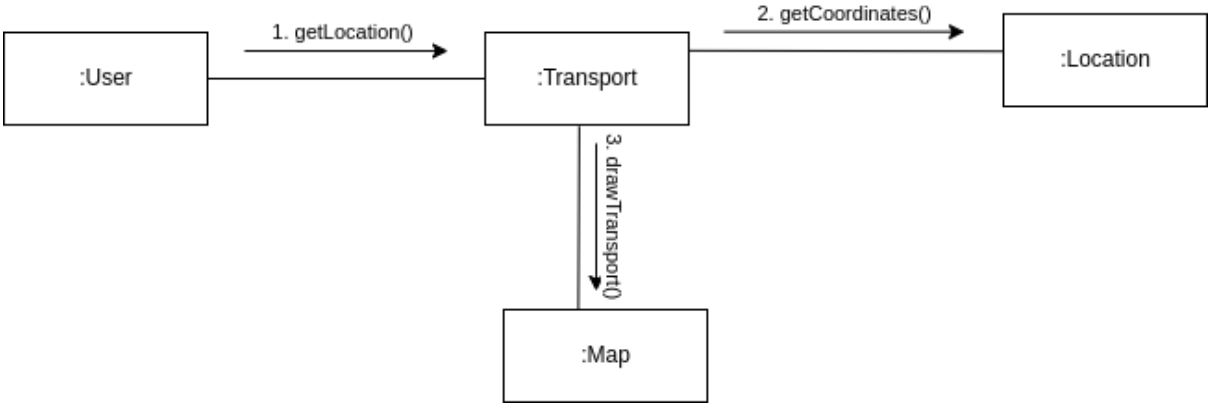

Рисунок 36. Диаграмма коммуникаций для просмотра местоположения транспортного средства.

Экземпляр User запрашивает местоположение транспорта методом getLocation() у экземпляра Transport. Транспортное средство методом getCoordinates() получает текущие координаты на карте мира и передаёт их экземпляру класса Map, который отображает транспортное средство на карте мира, используя метод drawTransport().

3.8. Диаграмма компонентов

На рисунке 37 представлена диаграмма компонентов.

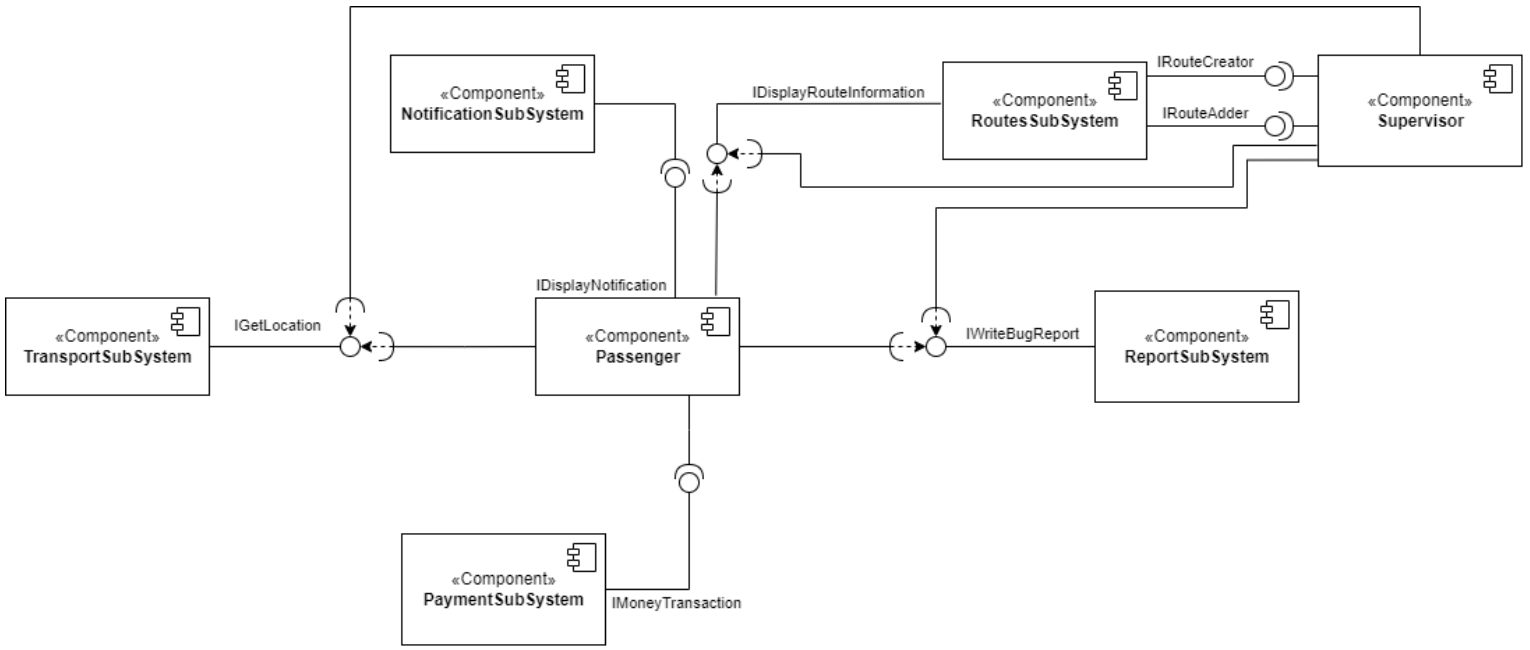

Рисунок 37. Диаграмма компонентов

На диаграмме компонентов представлены следующие компоненты:

1. **TransportSubSystem**

   Компонент, отвечающий за транспортную подсистему.

   Интерфейсы: IGetLocation — служит для отображения местоположения транспорта для Passenger.

   Состав: Transport, Location, Map.

2. **Passenger**

   Компонент, отвечающий за подсистему пассажира.

   Интерфейсы: IDisplayNotification — для работы с уведомлениями для пользователя.

   Состав: Passenger, Ticket, Trip, Login, Registration.

3. **NotificationSubSystem**

   Компонент, отвечающий за подсистему уведомлений.

   Состав: NotificationSystem, Notification, BillNotification.

4. **PaymentSubSystem**

   Компонент, отвечающий за подсистему оплаты.

   Интерфейс: IMoneyTransaction — проведение денежных транзакций для Passenger.

   Состав: PaymentSystem, Request, Transaction, Bill.

5. **RoutesSubSystem**

   Компонент, отвечающий за подсистему расписания.

   Интерфейс: IDisplayRouteInformation — отображение информации о расписании. IRouteCreator, IRouteAdder — интерфейсы для взаимодействия Supervisor для создания и редактирования расписания или маршрута.

   Состав: Route, Schedule, Stop.

6. **ReportSubSystem**

   Компонент, отвечающий за подсистему отчётов об ошибках.

   Интерфейс: IWriteBugReport — взаимодействие Passenger и Supervisor с системой обработки ошибок.

   Состав: Report, ReportSystem, Support.

7. **Supervisor**

   Компонент, отвечающий за подсистему для сотрудников депо.

   Состав: Supervisor, SupervisorRegistration.

3.9. Диаграмма развертывания

На рисунке 38 представлена диаграмма развертывания.

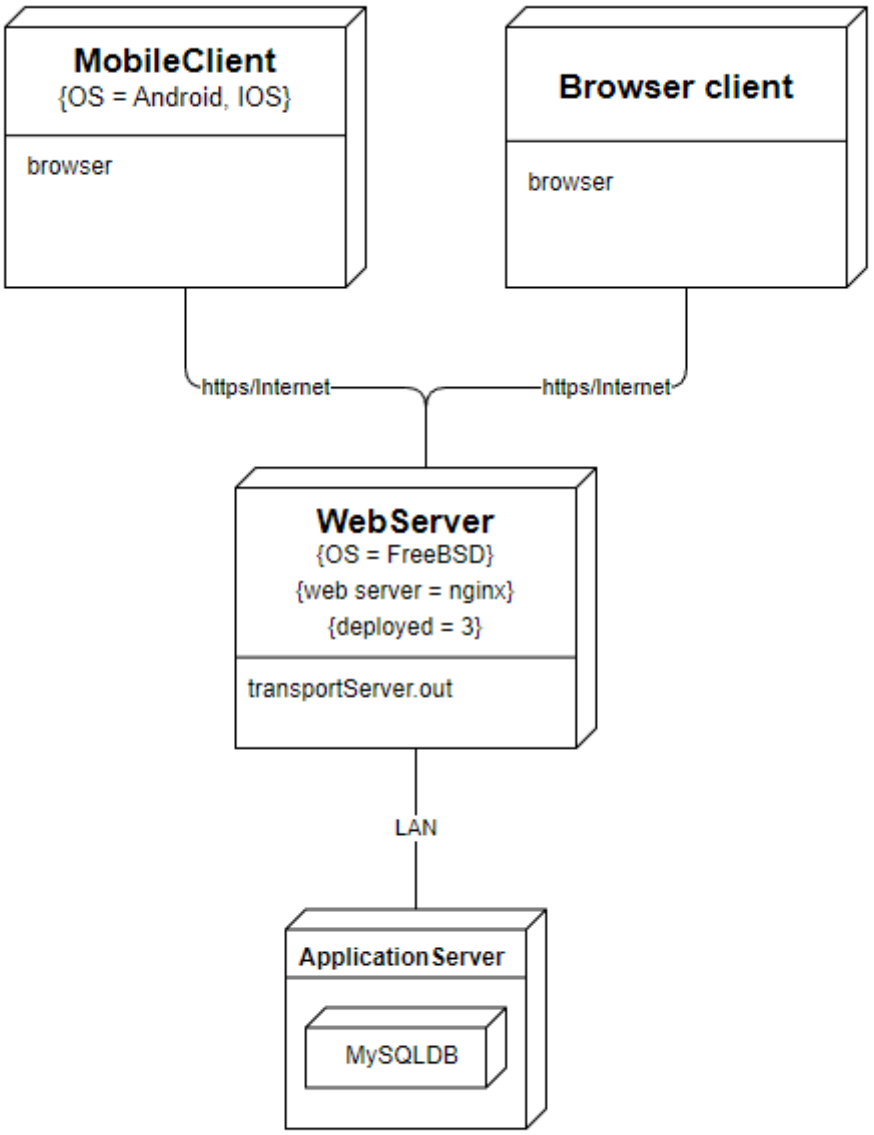

Рисунок 38. Диаграмма развертывания.

Browser client и Mobile client- клиенты, которые могут являться как пользователями, так и сотрудниками системы. Такое разделение отображает возможность входа в систему с разных типов устройств. Пользователи взаимодействуют с Web Server через интернет, в качестве

самого сервера используется сервер на платформе x86 FreeBSD на основе nginx из-за повышенной отказоустойчивости и недорогой эксплуатации. Web Server поддерживает интерфейсы программы для связи с основной логикой в Application Server, в котором также хранится база данных системы. Application Server в качестве базы данных использует MySQL DB.

3.10. Диаграмма пакетов

На рисунке 39 представлена диаграмма пакетов.

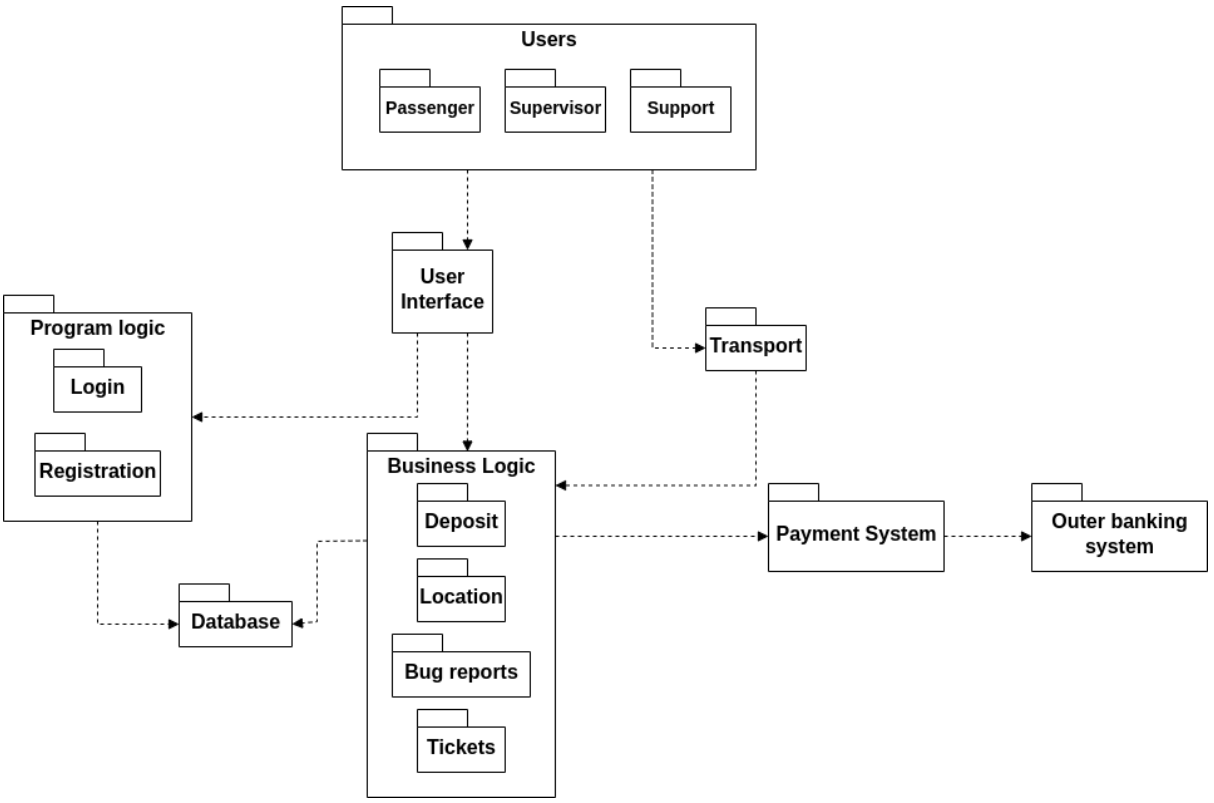

Рисунок 39. Диаграмма пакетов.

Users - Характеризует пользователей данного сервиса. User Interface служит для отправки информации пользователей к бизнес логике и наоборот. Таким же образом осуществляется связь с программной логикой приложения.

Transport - транспорт с названием, текущим маршрутом и расписанием.

Database - база данных, хранящая всю необходимую информацию о системе.

Outer Banking System - независимая внешняя банковская система, обслуживающая денежные транзакции.

Program logic - Техническая реализация пользовательских интерфейсов, также осуществляющая прямое взаимодействие с базой данных через авторизацию и регистрацию.

Business logic - обрабатывает поступающую от пользователей информацию, и с помощью полученных результатов, обращается к другим частям системы.

Payment system - взаимодействует с информацией о количестве денежных средств пассажира в системе.

4. СООТВЕТСТВИЕ СИСТЕМЫ ТРЕБОВАНИЯМ ТЗ

4.1. Соответствие требованиям

В таблице 1 представлены требования ТЗ и решения, удовлетворяющие эти требования.

ТРЕБОВАНИЯ ТЗ	РЕШЕНИЯ
Модуль пассажира должен предоставлять следующие функции: 1. Авторизация в системе. 2. Функция пополнения личного счёта. 3. Функция вывода средств. 4. Функция оплаты билета, проездного. 5. Функция просмотра транспорта на карте. 6. Функция просмотра времени прибытия для выбранной остановки. 7. Функция подачи жалобы на некорректную работу веб-приложения.	Авторизация в системе (1) осуществляется при помощи метода loginUser() в классе Login. Пополнение личного счёта (2) осуществляется при помощи метода depositMoney() в классе Passenger, вывод средств с личного счёта (3) - при помощи метода withdrawMoney() в Passenger. Оплата билета (4) производится при помощи метода buyTicket() в классе Passenger. Просмотр транспорта на карте (5) происходит с помощью метода drawTransport() в классе Map. Просмотр времени прибытия для выбранной остановки (6) происходит путём обращения к полю arrivals класса Schedule. Подача жалобы на некорректную работу приложения (7) производится при помощи метода bugReport() класса User, который

	является обобщением класса Passenger.
Модуль диспетчера должен предоставлять следующие функции: 1. Функция просмотра информации о транспорте на линиях, а также их местоположения. 2. Функция просмотра и редактирования расписания для каждой транспортной единицы (тип транспорта, время выезда на линию, плановый интервал движения между остановками). 3. Функция просмотра информации по выбранному маршруту с показом информации о пассажиропотоке, количестве купленных билетов. 4. Функция создания и редактирования маршрутов.	Просмотр информации о транспорте на линиях (1) осуществляется через обращение к полю transportOnLine класса Route, информации о местоположении - методом getCoordinates() класса Location. Просмотр и редактирование расписания (2) осуществляется путём обращения к полям stops/arrivals и методам addStop() / addArrival() класса Schedule. Просмотр информации по выбранному маршруту (3) осуществляется при помощи обращения к полям cost и transportOnLine класса Route. Создание и редактирование маршрутов (4) осуществляется через изменение полей cost, transportOnLine класса Route.
Модуль сотрудника технической поддержки должен предоставлять	Просмотр текущих жалоб от пользователей (1) осуществляется

следующие функции: 1. Функция просмотра текущих жалоб от пользователей. 2. Функция обратной связи с пользователями.	при помощи метода sendReportToSupport() класса ReportSystem. Обратная связь с пользователями осуществляется (2) путём ответа на жалобу с помощью метода replyReport() класса Support.

ЗАКЛЮЧЕНИЕ

Был написан проект, представляющий собой описание архитектуры автоматизированной информационной системы ИС СКТ.