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

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту
по дисциплине «Спецификация, проектирование и архитектура программных систем»
Тема: Автоматизированная система управления стадионом

Студенты гр. 9303 _____ Низовцов Р.С.
_____ Муратов Р.А.
_____ Ахримов А.М.
_____ Зарезина Е.А.
Преподаватель _____ Романенко С.А.

Санкт-Петербург
2021

СОДЕРЖАНИЕ

1	Общие положения	5
1.1	Полное наименование проектируемой автоматизированной системы и наименования документов, их номера и дата утверждения, на основании которых ведется проектирование АС	5
1.2	Перечень организаций, участвующих в разработке системы, сроки выполнения работ	5
1.3	Цели, назначение и области использования АС	5
1.4	Сведения об использованных при проектировании нормативно-технических документах	6
1.5	Очередность создания системы и объём каждой очереди	6
1.5.1	Проектирование. Разработка эскизного проекта и технического задания	6
1.5.2	Разработка программного обеспечения для реализации функций системы. Разработка рабочей документации. Отладка программных модулей	7
1.5.3	Проведение опытной эксплуатации. Подготовка персонала. Проведение испытаний. Ввод в действие.	7
1.6	Словарь терминов	7
2	Описание процесса деятельности	10
2.1	Описание технологических процессов предметной области	10
2.1.1.	Контекстная диаграмма IDEF0	10
2.1.2.	Диаграмма IDEF0 1-го уровня (A0)	11
2.1.3.	Диаграмма IDEF0 2-го уровня, детализирующая процесс обработки на стороне сайта стадиона (A1)	12
2.1.4.	Диаграмма IDEF0 3-го уровня, детализирующая процесс авторизации пользователя (A11)	14
2.1.5.	Диаграмма IDEF0 2-го уровня, детализирующая процесс покупки и возврата билета (A2).	15
2.2	Описание потока данных предметной области	16
2.2.1.	Диаграмма DFD	16
2.3	Описание информационного обеспечения	18
2.3.1	ER-диаграмма	18

3 Основные технические решения 21
3.1 Состав функций, комплексов задач, реализуемых системой 21
3.1.1 UML диаграмма сценариев использования 21
3.1.1.1 Подсистема «Оплата» 21
3.1.1.2 Подсистема «Аренда» 22
3.1.1.3 Подсистема «Пользователь» 23
3.1.1.4 Взаимодействие с системой при посещении стадиона 24
3.2 Решения по структуре системы, подсистем. Решения по взаимосвязям АС со смежными системами, обеспечению ее совместимости 25
3.2.1 UML диаграмма классов 25
3.2.2 UML диаграмма объектов 28
3.2.3 UML диаграмма компонентов 30
3.2.4 UML диаграмма пакетов 31
3.3 Решения по средствам и способам связи для информационного обмена между компонентами системы 33
3.3.1 UML диаграмма последовательности процесса авторизации пользователя. 33
3.3.2 UML диаграмма последовательности процесса покупки билетов через сайт. 35
3.3.3 UML диаграмма последовательности процесса покупки билетов через терминал. 37
3.3.4 UML диаграмма коммуникации 39
3.4 Решения по входным и выходным документам и сообщениям, последовательности обработки информации, алгоритмам процедур и операции 40
3.4.1 UML диаграмма состояний мероприятия 40
3.4.2 UML диаграмма состояний покупки билетов 42
3.4.3 UML диаграмма деятельности для процесса покупки билета 44
3.4.4 UML диаграмма деятельности для аренды стадиона 46
3.5 Решения по комплексу технических средств, его размещению на объектах эксплуатации. 48
3.5.1 UML диаграмма развертывания 48
3.6 Решения по составу программных средств, языкам и технологиям реализации 49

3.7 Перечень и описание шаблонов проектирования, использованных при проектировании архитектуры 50
3.7.1 Шаблон проектирования для «расслоения» архитектуры системы 50
3.7.2 Шаблоны проектирования для представления бизнес-логики и работы с данными. 51
Заключение 53

1. ОБЩИЕ ПОЛОЖЕНИЯ.

1.1. Полное наименование проектируемой автоматизированной системы и наименования документов, их номера и дата утверждения, на основании которых ведется проектирование АС.

Автоматическая система управления стадионом.

Условное обозначение: АС управления стадионом.

Система создается на основании договора №9303 от 01.09.2021 между ООО «Rucky Crewky» и ООО «Зенит-Арена».

1.2. Перечень организаций, участвующих в разработке системы, сроки выполнения работ.

Систему разрабатывает ООО «Rucky Crewky».

Срок начала работ: 08.10.2021

Срок окончания работ: 10.12.2021

1.3. Цели, назначение и области использования АС.

АС управления стадионом предназначена для повышения оперативности и качества принимаемых управленческих решений сотрудниками Заказчика. АС управления стадионом включает в себя разработку системы управления над:

- Системой безопасности
- Средствами покупки и возврата билетов
- Сайтом для дистанционной работы с персоналом, арендаторами (организациями, арендующими площадку стадиона) и покупателями

Целями создания АС управления стадионом являются:

- Уменьшение времени реакции на чрезвычайные ситуации в 1.5 – 1.7 раза
- Уменьшение количества чрезвычайных ситуаций на 30%

- Уменьшение количества посредников при передаче указаний между персоналом и командой управления
- Добавление новых способов покупки билетов
- Добавление дистанционного взаимодействия между Заказчиком и арендаторами
- Формирование любой отчётности не должно требовать ручных операций.

1.4. Сведения об использованных при проектировании нормативно-технических документах.

Документирование проекта автоматизации регламентируется следующими стандартами:

- ГОСТ 34.601-90 Автоматизированные системы. Стадии создания.
- ГОСТ 34.602-89 Техническое задание на создание автоматизированной системы.
- РД 50-34.698-90 Автоматизированные системы. Требование к содержанию документов.

1.5. Очередность создания системы и объём каждой очереди.

1.5.1. Проектирование. Разработка эскизного проекта и технического задания (3 месяца).

1. Формализация и спецификация требований к автоматизированной системе, составление и согласование технического задания.
2. Анализ предметной области, составление словаря терминов, IDEF0 и DFD диаграмм и модели предметной области.
3. Функциональное проектирование, составление модели вариантов использования в виде Use Case диаграммы.
4. Проектирование архитектуры Системы, составление UML диаграмм с описанием.

1.5.2. Разработка программного обеспечения для реализации функций системы. Разработка рабочей документации. Отладка программных модулей. (1.5 месяца).

1. Разработка программного обеспечения с последующим предоставлением выполняемых программных модулей.
2. Составление рабочей документации на автоматизированную систему.
3. Проведение тестирования и доработка программного обеспечения по предоставленным Заказчиком замечаниям и предложениям.

1.5.3. Проведение опытной эксплуатации. Подготовка персонала. Проведение испытаний. Ввод в действие. (1 месяц).

1. Составление акта приёмки Системы в опытную эксплуатацию.
2. Проведение испытаний и составление протокола испытаний.
3. Составление акта приёмки Системы в эксплуатацию.
4. Составление акта завершения работ

1.6. Словарь терминов.

Авторизация	Процесс проверки и подтверждения данных при попытке войти в систему
Администратор	Лицо, управляющее командой персонала стадиона
Арендатор	Физическое или юридическое лицо, получившее стадион во временное пользование и заключившее договор с владельцем стадиона
Аутентификация	Процедура проверки подлинности введённых данных при авторизации
БД	Организованная совокупность данных, необходимая для

	обеспечения работоспособности спортивного объекта
Билет	Документ, удостоверяющий право прохода на стадион в целях посещения спортивно-массового мероприятия
Данные	Совокупность значений, содержащих информацию, приемлемую для хранения и обработки системой
ЕЦУС	Единый центр управления стадионом
Журналирование	Форма автоматической записи в хронологическом порядке операций в системе
Инвентарь	Совокупность различных предметов хозяйственного обихода или производственного назначения, необходимая для осуществления и поддержания работоспособности стадиона
Персонал	Постоянный состав работников стадиона
Полномочия	Официально предоставленные сотруднику права и обязанности
Посетитель	Лицо, приходящее на стадион с целью просмотра спортивного мероприятия
Ремонт	Комплекс мероприятий по восстановлению работоспособности стадиона
Системный администратор	Лицо, осуществляющее обеспечение корректной работы ПО стадиона
Стадион	Сооружение для проведения спортивно-массовых мероприятий
Техническое обслуживание	Комплекс мероприятий по поддержанию работоспособности стадиона
Терминал	Электронное устройство, предназначенное либо для проверки билетов на подлинность, либо для покупки билетов

Карта	Визуализация пространства стадиона с условными обозначениями
Расписание	Упорядоченный по датам список мероприятий
Договор	Соглашение двух или нескольких сторон о взаимных правах и обязательствах

2. ОПИСАНИЕ ПРОЦЕССА ДЕЯТЕЛЬНОСТИ.

2.1. Описание технологических процессов предметной области

2.1.1. Контекстная диаграмма IDEF0 (A-0).

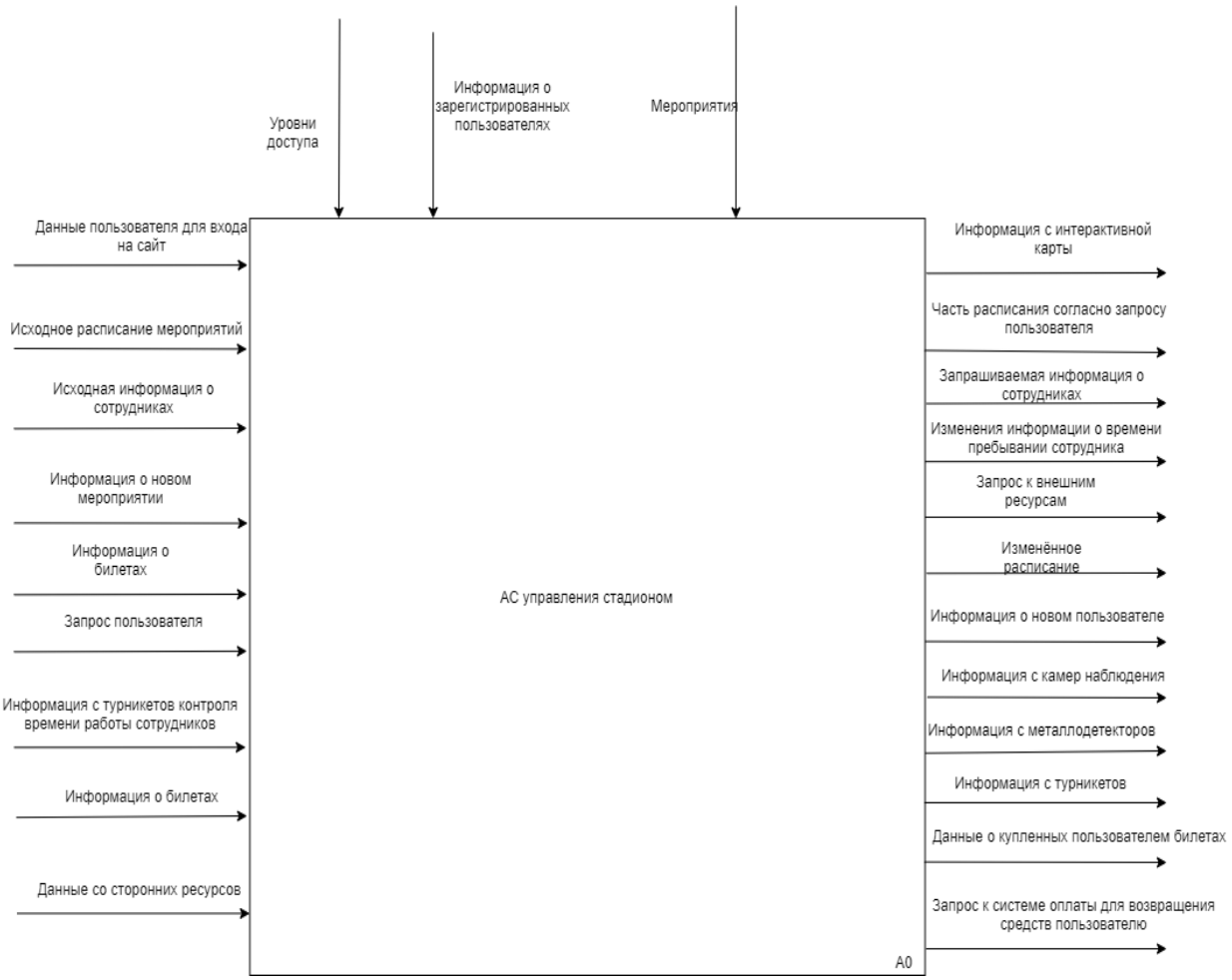

Рисунок 1 – Диаграмма IDEF0

Основная диаграмма, представляет собой всю функциональность системы в целом. Запрос пользователя представляет собой обобщенные данные, которые будут расслаиваться по мере детализации системы.

2.1.2. Диаграмма IDEF0 1-го уровня (A0).

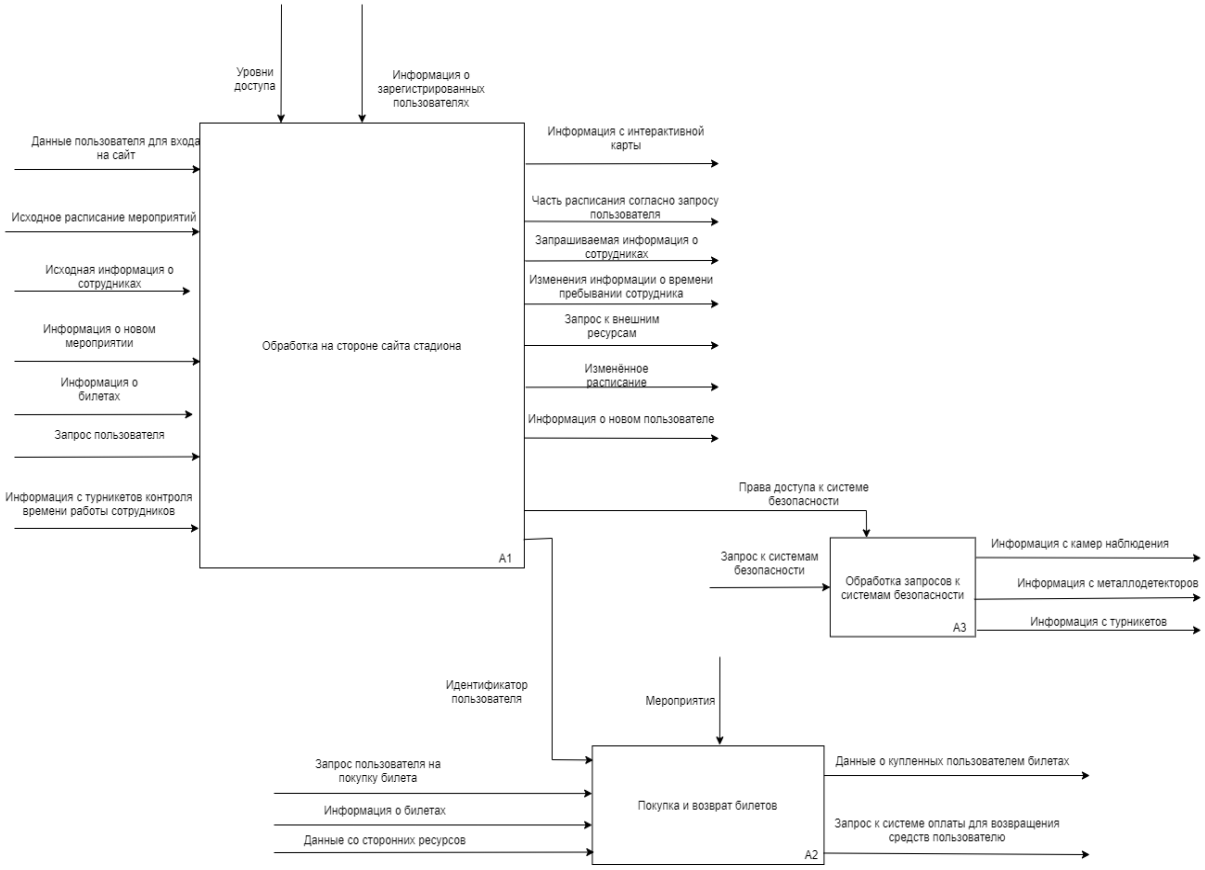

Рисунок 2 – Диаграмма IDEF0 1-го уровня (A0)

На данном уровне детализации выделяются три основные функциональные системы в соответствии с техническим заданием. Большинство взаимодействий с системой начинается с сайта. Без авторизации пользователя на сайте система не даст доступ к системам безопасности. Покупку билетов пользователь может совершить на сайте стадиона, на территории стадиона и с помощью специальных сайтов по покупки билетов.

2.1.3. Диаграмма IDEF0 2-го уровня, детализирующая процесс обработки на стороне сайта стадиона (A1).

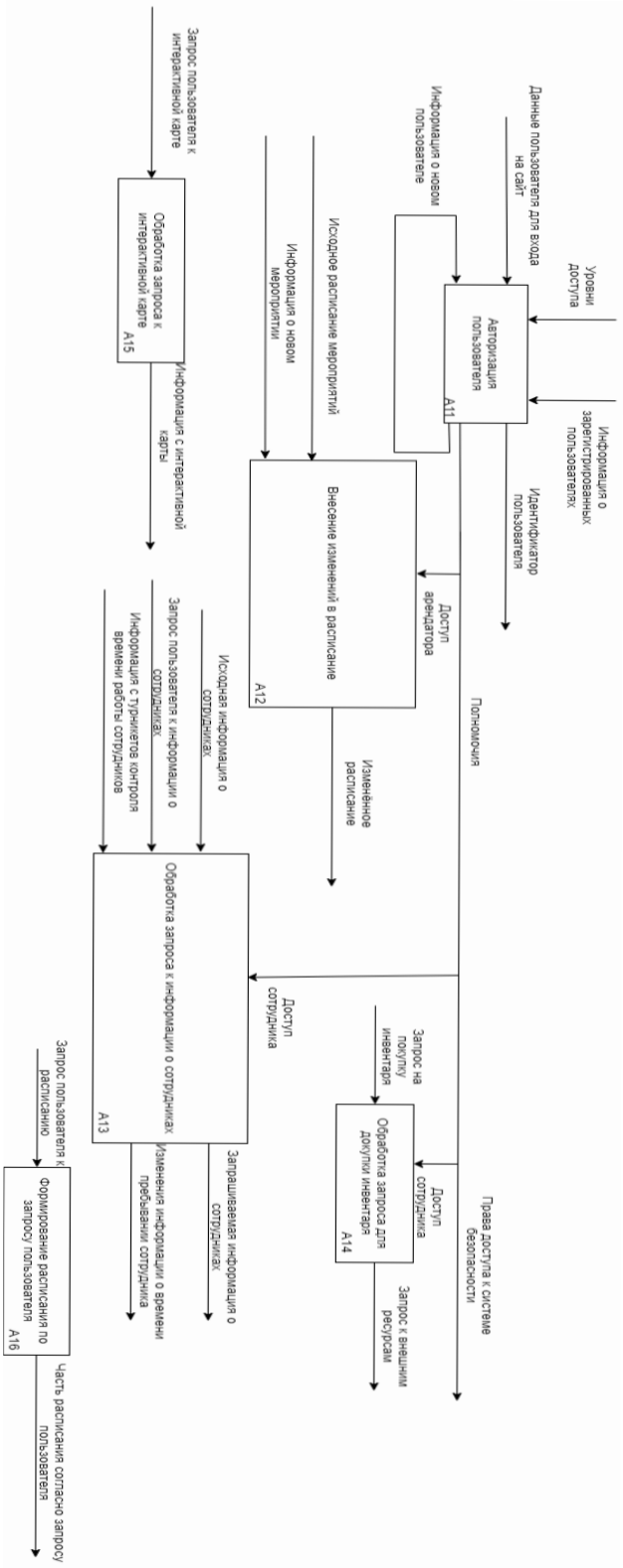

Рисунок 3 – Диаграмма IDEF0 2-го уровня (A1)

На данной диаграмме показана функциональность сайта. Для некоторых функций сайта пользователю необходимо подтвердить свой уровень доступа. Например, чтобы сделать запрос на покупку инвентаря, нужно иметь доступ сотрудника. Некоторые функции такие, как просмотр интерактивной карты или просмотр расписания доступны любому посетителю сайта.

2.1.4. Диаграмма IDEF0 3-го уровня, детализирующая процесс авторизации пользователя (A11).

Рисунок 4 – Диаграмма IDEF0 3-го уровня (A11)

На данной диаграмме показан процесс авторизации пользователя. Если пользователь не авторизован, ему предлагается зарегистрироваться. Если пользователь существуют, система определяет категорию пользователя и выдаёт ему соответствующее полномочия.

2.1.5. Диаграмма IDEF0 2-го уровня, детализирующая процесс покупки и возврата билета (A2).

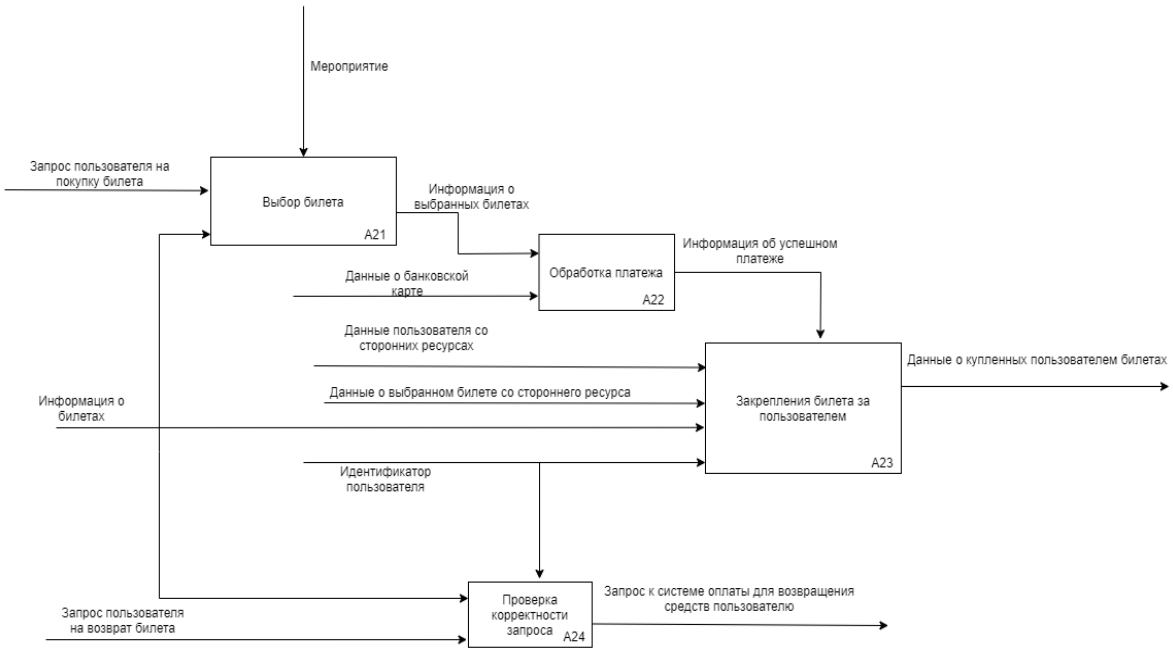

Рисунок 5 -- Диаграмма IDEF0 2-го уровня (A2)

На данной диаграмме показан процесс покупки билета. Билет закрепиться за пользователем, если после выбора билета успешно пройдёт проверка платежа. На билете будет идентификатор пользователя. Если пользователь покупает билет в терминале на территории стадиона, будет указан идентификатор терминала. Если пользователь покупает билет со сторонних ресурсов, то идентификатор сформируется на основе данных, пришедших с этих ресурсов.

2.2. Описание потока данных предметной области.

2.2.1. Диаграмма DFD

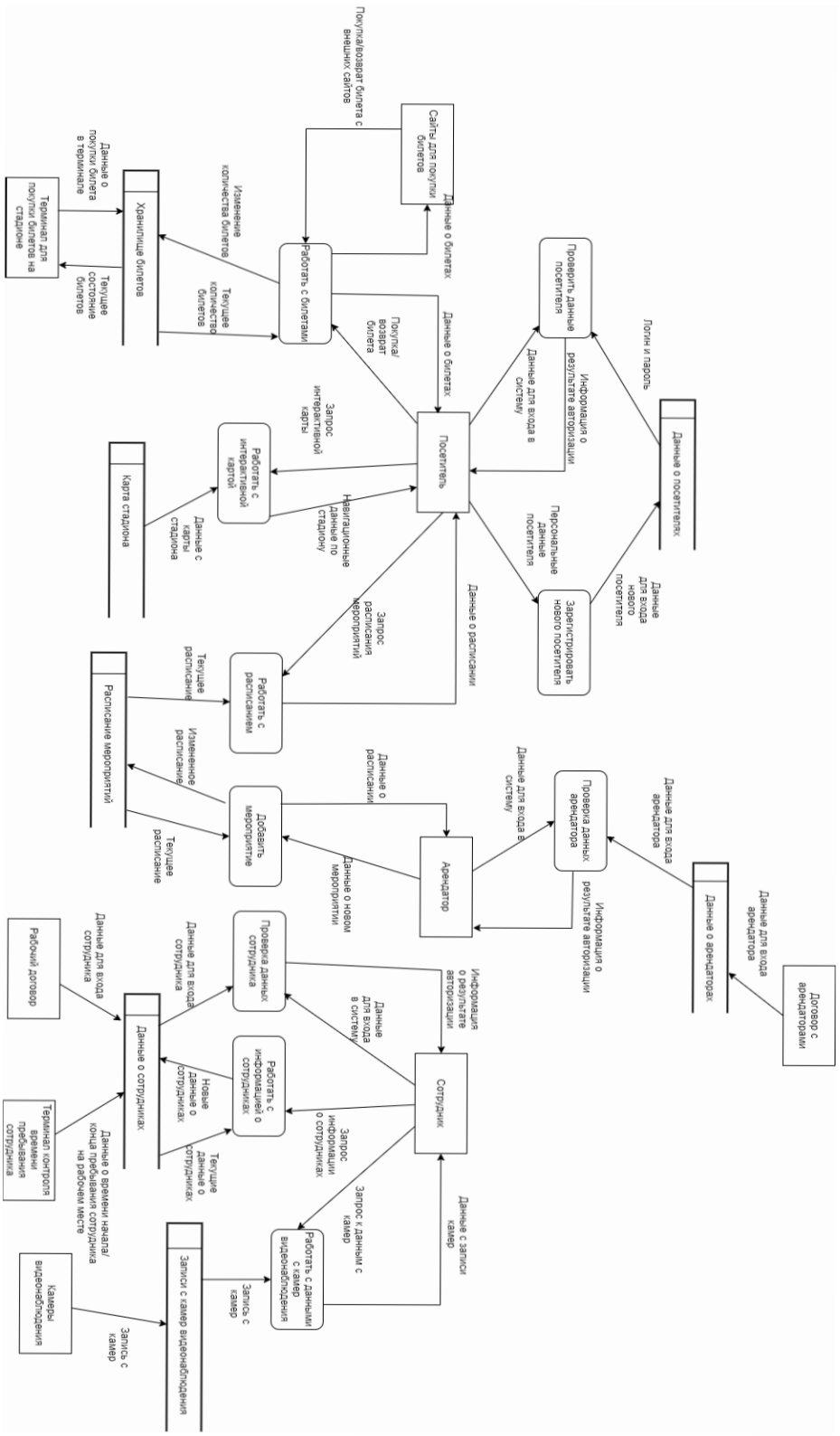

Рисунок 6 – Диаграмма DFD

На данной диаграмме показаны потоки движения информации между хранилищами данных и процессами. У категории пользователей «Посетитель» есть возможность самостоятельно зарегистрироваться в системе. Арендатору необходимо сначала заключить с администрацией стадиона договор аренды, а сотруднику – рабочий договор. Взаимодействовать с билетами могут такие сущности, как «Посетитель», «Сайты для покупки билетов» и «Терминал для покупки билетов на стадионе». «Посетитель» может только просматривать расписание мероприятий, когда как «Арендатор» может внести изменения в него. «Сотрудник» может посмотреть информацию о сотрудниках или внести изменения. Также ему доступны записи с камер видеонаблюдения. Сущность «Терминал контроля времени пребывания сотрудника» вносит данные о времени пребывания сотрудника на рабочем месте.

2.3. Описание информационного обеспечения.

2.3.1. ER-диаграмма.

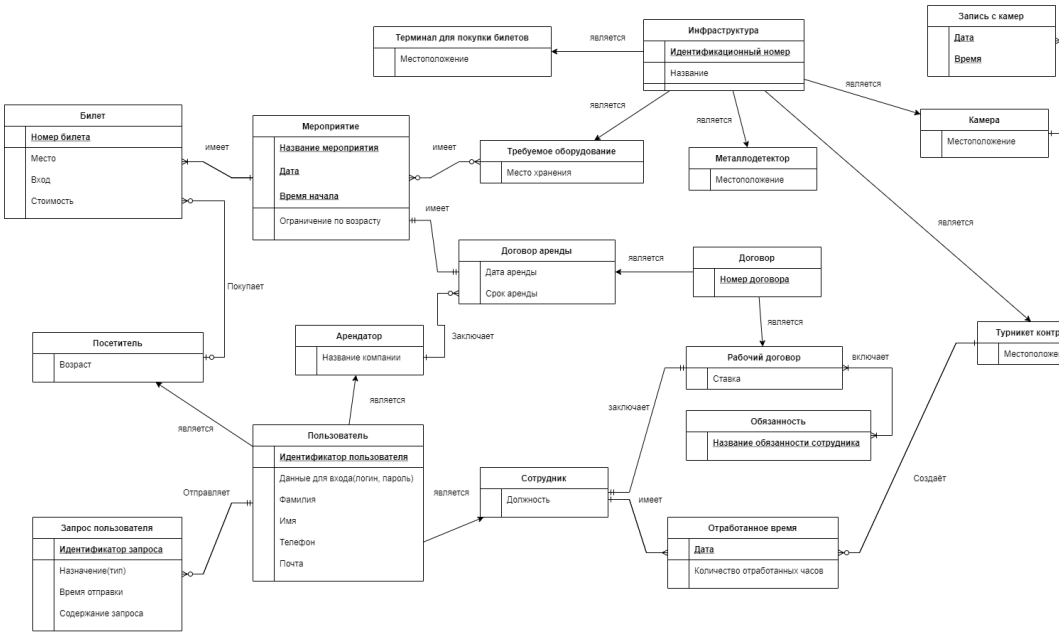

Рисунок 7 – ER-диаграмма

- Сущность «Билет» представляет собой электронный или физический документ, удостоверяющий право посетить спортивно-культурное мероприятие. Билет однозначно идентифицируется своим номером, в билете содержится информация о месте посетителя на стадионе (либо ряд и номер кресла, либо место относительно сцены); стоимости билета и входе, через который посетитель должен будет пройти, чтобы добраться до своего места. Билет дает доступ только к одному мероприятию и им может владеть только один посетитель.
- Сущность «Пользователь» - общая сущность, которая представляет собой пользователя сайта. Он однозначно определяется идентификатор, который выдается в зависимости от типа пользователя (см. «Посетитель», «Сотрудник», «Арендатор»). Также имеется фамилия, имя, телефон и адрес электронной почты.
- Сущность «Посетитель» - категория сущности «Пользователь» и представляет собой человека, который намеревается посетить мероприятие. Возраст — единственный атрибут сущности. Посетитель может иметь несколько билетов на различные мероприятия.

- Сущность «Сотрудник» – категория сущности «Пользователь» и представляет собой физическое лицо, официально заключившее трудовой договор с администрацией стадиона. В качестве информации о сотруднике указывается его должность. Сотрудник имеет одну или более обязанностей и несколько записей об отработанном времени.
- Сущность «Арендатор» – категория сущности «Пользователь» и представляет собой юридическое или физическое лицо, заключившее с администрацией стадиона договор об аренде стадиона на определенный промежуток времени. В качестве атрибута используется название компании, арендующее стадион (опциональный атрибут).
- Сущность «Мероприятие» – культурно-спортивное мероприятие, проводимое на стадионе. Однозначно определяется названием события и временем проведения (дата + время). Также имеется ограничение по возрасту. На мероприятие может быть один или несколько билетов, оно имеет одного арендатора-организатора и для его проведения от стадиона требуется какое-то оборудование (0 или более).
- Сущность «Инфраструктура» - общая сущность, представляющая собой элемент инфраструктуры, которым обладает стадион. В качестве первичного ключа используется идентификационный номер. Также указывается название элемента.
- Сущность «Требуемое оборудование» - категория сущности «Инфраструктура», которая представляет собой оборудование, которое может быть взято в аренду у стадиона для проведения мероприятия. Атрибут — место хранения оборудования. Оборудование может быть использовано на нескольких мероприятиях или вообще не использоваться.
- Сущность «Металлодетектор» - категория сущности «Инфраструктура», представляет собой оборудование для проверки посетителей. Имеет атрибут местоположение.
- Сущность «Турникет контроля времени» - категория сущности «Инфраструктура», представляет собой оборудование для контроля

времени пребывания сотрудников на рабочем месте. Имеется атрибут местоположение. Создаёт записи сущности «Отработанное время».

- Сущность «Терминал для покупки билетов» - категория сущности «Инфраструктура», представляет собой оборудование для покупки билетов на территории стадиона. Имеется атрибут местоположение.
- Сущность «Камера» – категория сущности «Инфраструктура» и представляет собой камеру видео-наблюдения. Хранится информация о её местоположении. У камеры может быть одна или несколько записей.
- Сущность «Запись с камер» - запись с камеры видео-наблюдения. Используется составной ключ: дата + время. Запись относится только к одной камере.
- Сущность «Отработанное время» - запись с информацией об отработанных сотрудником часах в определенную дату. Запись относится только к одному сотруднику.
- Сущность «Обязанность» представляет собой задачу, которую необходимо выполнить сотрудникам. Определяется названием задачи.
- Сущность “Договор” – общая сущность, представляет собой заключаемые договора. Однозначно определяется номером договора.
- Сущность “Рабочий договор” представляет собой договор, заключенный между администрацией стадиона и сотрудником. В договоре прописывается ставка сотрудника.
- Сущность “Договор аренды” представляет собой договор, заключенный между администрацией стадиона и арендатором. В договоре прописывается дата и срок аренды.
- Сущность «Запрос пользователя» представляет собой запросы, посылаемые пользователем при взаимодействии с сайтом.

3. ОСНОВНЫЕ ТЕХНИЧЕСКИЕ РЕШЕНИЯ.

3.1. Состав функций, комплексов задач, реализуемых системой.

3.1.1. UML диаграмма сценариев использования.

3.1.1.1. Подсистема «Оплата»

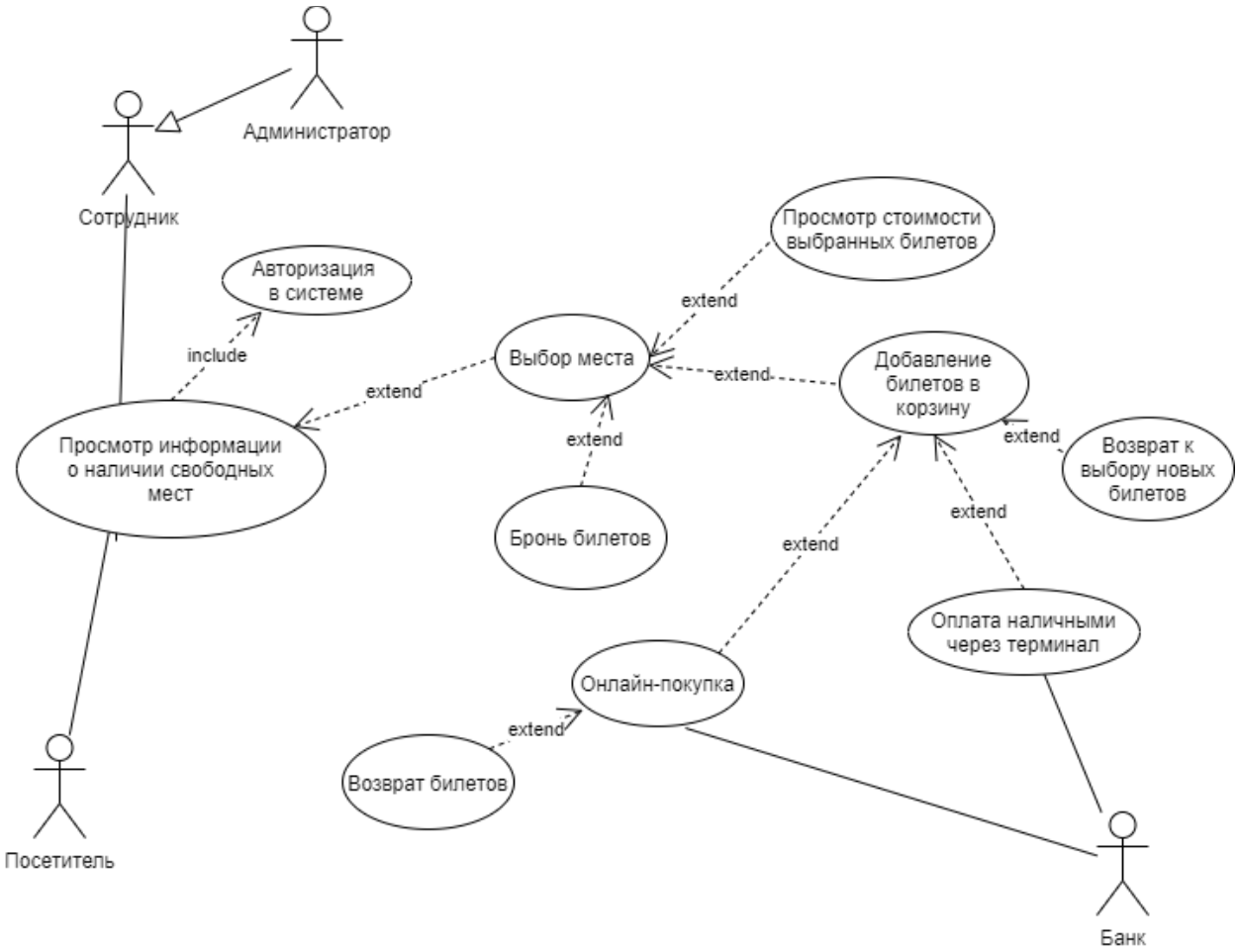

Рисунок 8 – Use case диаграмма подсистемы «Оплата»

На Use Case диаграмме представлено описание возможных сценариев работы с подсистемой «Оплата». Как сотрудник, так и посетитель может просмотреть информацию о наличии свободных мест. Это действие подразумевает собой предварительную авторизацию в системе. При просмотре данной информации в системе есть возможность выбрать место (действие связано отношением extend). Выбирая место, пользователь может просмотреть стоимость выбранных билетов, забронировать билеты, либо добавить их в корзину (все действия связаны отношением extend). При добавлении билетов в корзину пользователь может оплатить покупку через терминал, осуществить онлайн-покупку, а также вернуться к предыдущим страницам для выбора дополнительных билетов (все действия связаны отношением extend). При покупке билетов онлайн можно оформить возврат билетов при необходимости (все действия связаны отношением extend).

3.1.1.2. **Подсистема «Аренда»**

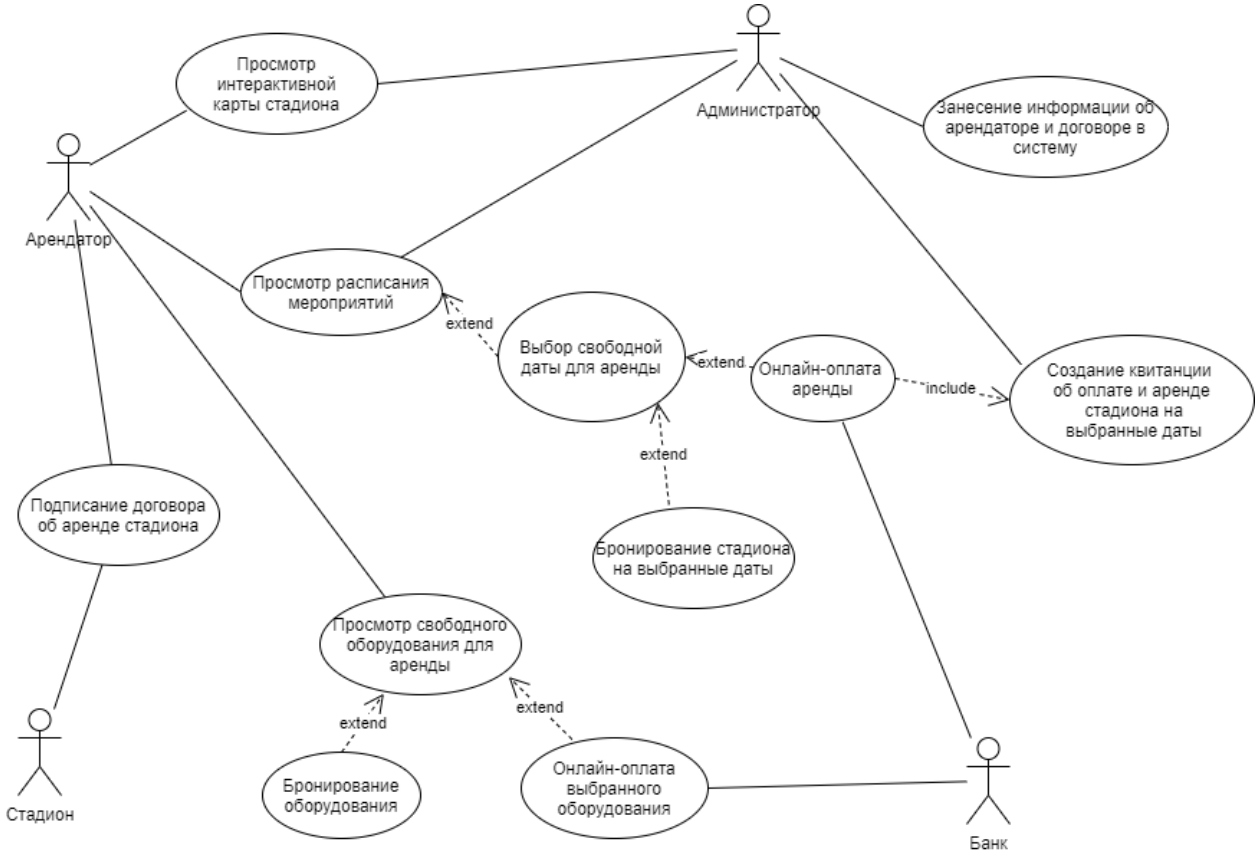

Рисунок 9 – Use case диаграмма подсистемы «Аренда»

На Use Case диаграмме представлено описание возможных сценариев работы с подсистемой «Аренда». Арендатор может просмотреть интерактивную карту стадиона, просмотреть расписание мероприятий (эти действия также доступны и Администратору системы), подписать договор об аренде стадиона, а также посмотреть свободное оборудование для аренды. При просмотре расписания мероприятий у арендатора и администратора есть возможность выбрать свободную дату для аренды (действие связано отношением extend). После этого действия возможно бронирование стадиона на выбранные даты (действие связано отношением extend), а также онлайн-оплата аренды, включающая в себя создание администратором квитанции об оплате и аренде стадиона на выбранные даты (действие связано отношением include). При просмотре арендатором свободного оборудования для аренды возможно бронирование данного оборудования, а также онлайн-оплата аренды выбранного оборудования (все действия связаны отношением extend).

3.1.1.3. **Подсистема «Пользователь»**

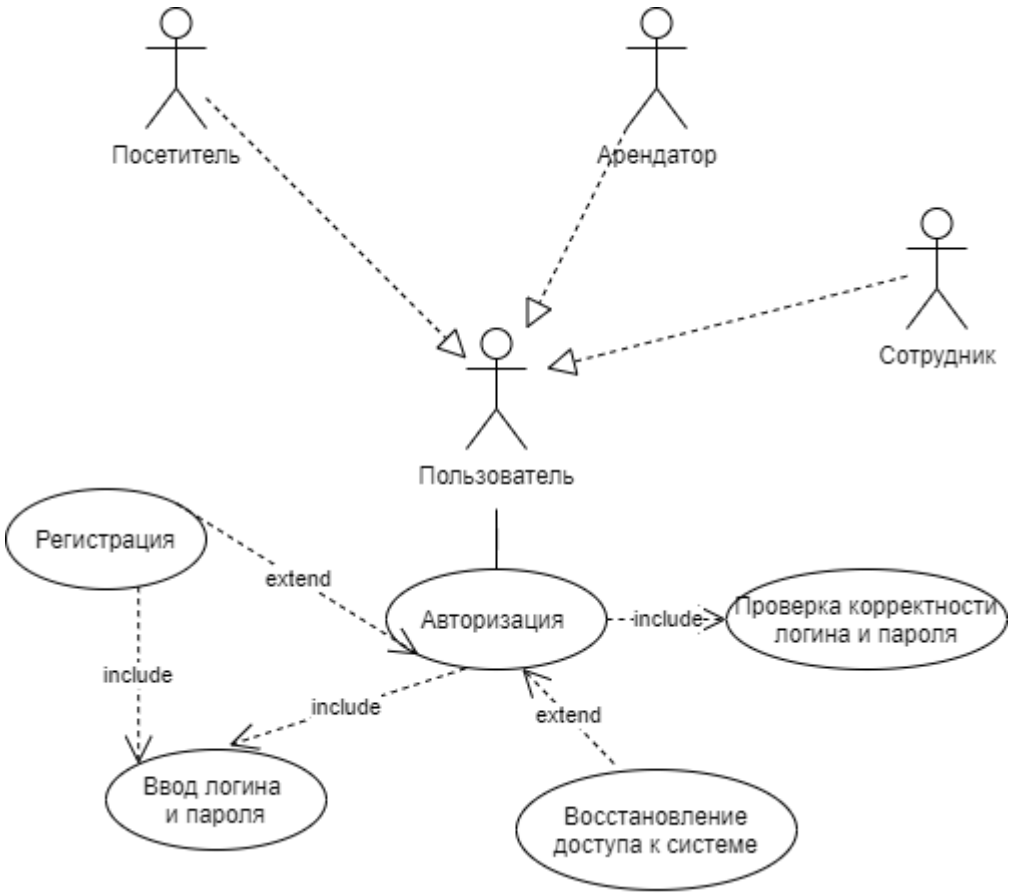

Рисунок 10 – Use case диаграмма подсистемы «Пользователь»

На Use Case диаграмме представлено описание возможных сценариев работы с подсистемой «Пользователь». Пользователями могут быть как посетитель, так арендатор или сотрудник стадиона. Пользователю необходимо произвести авторизацию, которая включает в себя проверку корректности логина и пароля (действие связано отношением include). Если пользователь в системе ещё не зарегистрирован, ему предоставляется возможность пройти регистрацию (действие связано отношением extend), при этом и авторизация, и регистрация подразумевают под собой ввод логина и пароля (действия связаны отношением include). Если пользователь не может войти в систему по причине того, что он забыл логин или пароль, то при авторизации есть возможность восстановить доступ к системе.

3.1.1.4. **Взаимодействие с системой при посещении стадиона**

Рисунок 11 – Use case диаграмма взаимодействия с системой при посещении стадиона

На Use Case диаграмме представлено описание возможных сценариев работы с системой при посещении стадиона. При посещении стадиона сотрудник стадиона и администратор системы осуществляют фиксацию времени начала и окончания рабочей смены сотрудника. Арендатор может подписать договор об аренде. Посетитель, администратор и сотрудник стадиона могут купить билеты на следующее мероприятие. При посещении стадиона посетителем сотрудник стадиона проверяет билеты на подлинность. Все акторы имеют возможность просмотреть интерактивную карту стадиона.

3.2. Решения по структуре системы, подсистем. Решения по взаимосвязям АС со смежными системами, обеспечению ее совместимости.

3.2.1. UML диаграмма классов.

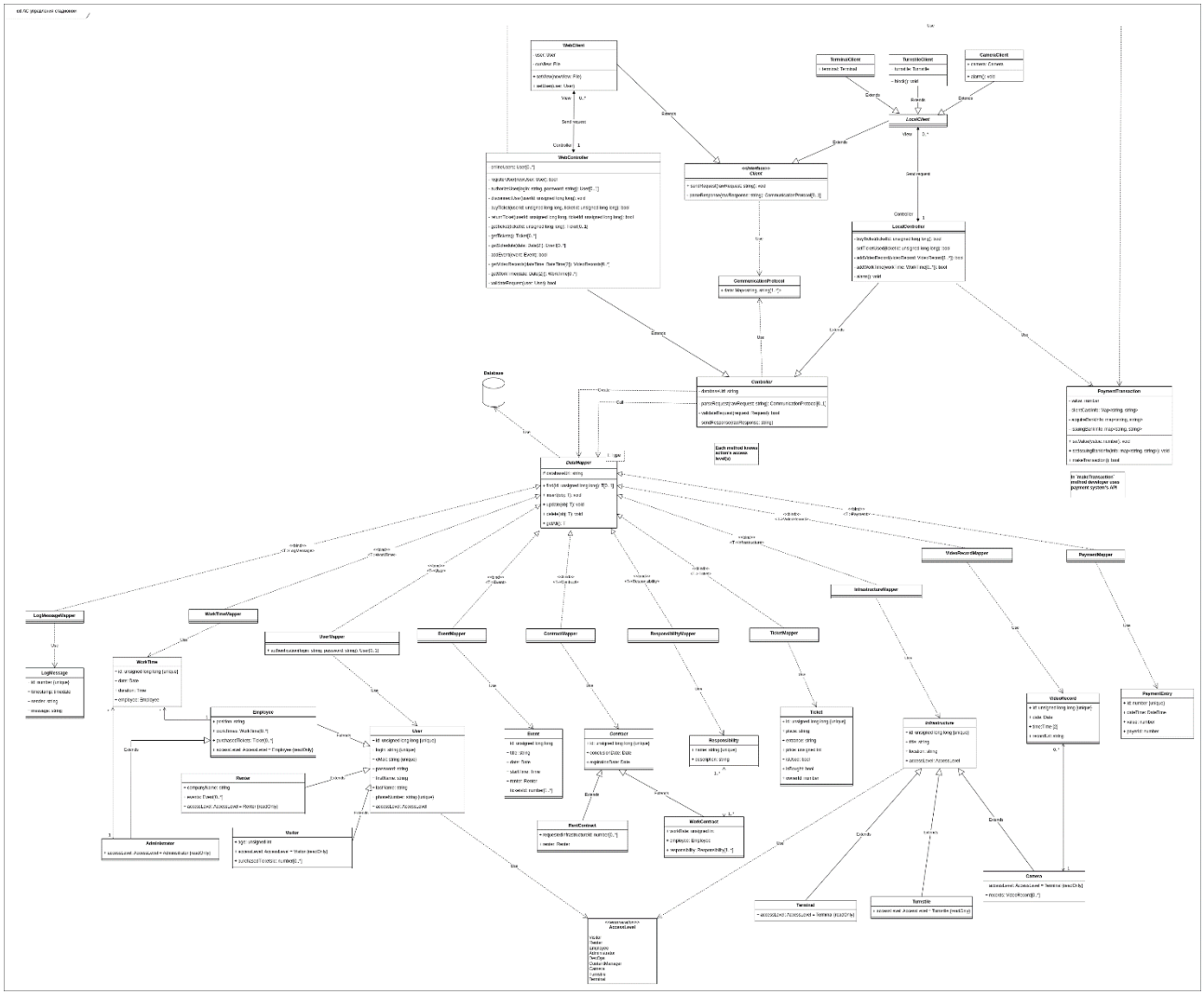

Рисунок 12 – UML диаграмма классов

При создании диаграммы классов использовалась концепция Model-View-Controller (MVC).

View:

- WebClient — класс, представляющий собой сущность, которая будет находится в бразуере пользователя, либо с которой будет взаимодействовать сторонние сервисы для покупки билетов.

- TerminalClient, TunstileClient, CameraClient — классы, представляющие собой терминал, турникет и камеру соответственно, которые подключены к локальной сети и могут взаимодействовать с контроллером.
- Интерфейс Client позволяет отправлять запросы по локальной сети к контроллеру и парсить его ответы.

Controller:

- WebController, LocalController — контроллеры для запросов по сети Интернет и локальной сети соответственно. Разделение необходимо, так как функционал для локальных и сетевых клиентов отличается.
- Оба контроллера используют класс PaymentTransaction, который представляет собой банковскую транзакцию для оплаты билета.
- Абстрактный класс Controller предоставляет общий функционал контроллеров по отправке ответов, парсинга запросов и их валидации. Также задается URL для подключения к базе данных.

Model:

- Для каждой сущности из модели предметной области в базе данных созданы соответствующие классы: Ticket, Employee, Renter и т. д. (используется PoEAA-паттерн «Domain model»), для каждого из которых задан mapper, задача которого - инкапсулировать код для работы с самой базой данных. В данном случае использован промышленный паттерн проектирования «Data mapper».

Перечисление AccessLevel задает всевозможные уровни доступа, они используются в методах контроллеров для проверки,

может ли пользователь системы выполнять выбранную операцию (методы сами «знают», какие пользователи могут их выполнять).

Шаблон MVC использован для повышения повторного использования кода, который будет написан при реализации системы, т. е. чтобы в случае необходимости, можно было бы изменить View, не меняя контроллер, и наоборот.

В данной системе используется «простая» разновидность шаблона «Модель предметной области», так как одному объекту домена соответствует одна реляционная таблица. Использование данного шаблона позволяет упростить понимание диаграммы классов, причем «Модель предметной области» напоминает соответствующую базу данных. В качестве примера можно привести класс «Emploee» (Работник), который соответствует реляционной таблице, в которой имеются атрибуты id, login, password, eMail, position, workTimes и т. д.

Шаблон «Data mapper» позволяет упростить взаимодействие с БД и сделать его более гибким. Данный шаблон хорошо работает в связке с паттерном «Domain model». Рассмотрим, например, класс «Event» (Событие). Контроллер при необходимости создает соответствующий класс, в данном случае это EventMapper, который инкапсулирует в себе код обращения к БД и выполнения определенных операции (поиск, вставка, обновление, удаление). То есть классы модели предметной области хранят только данные и не знают о существовании БД. Для каждого события выпускается несколько билетов, в БД это будет отражаться в виде внешних ключей в реляционной таблице «Событие», который ссылаются на ключи из таблицы «Билет». В задачи «преобразователя» (mapper) будет входить нахождение всех билетов, на которые ссылается определенное событие, и предоставление этой информации в виде поля-массива (в данном примере это ticketsId).

3.2.2. UML диаграмма объектов.

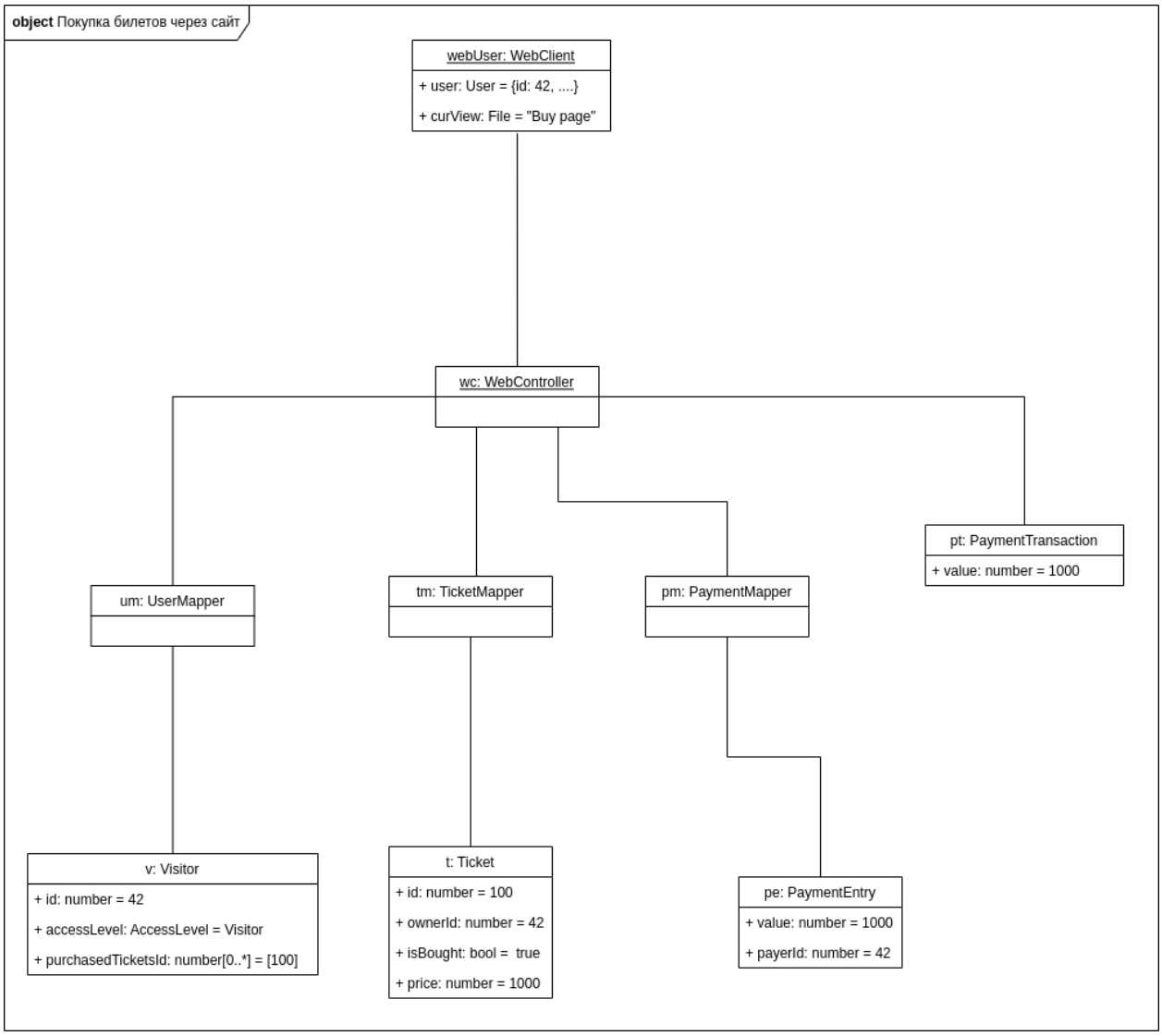

Рисунок 13 – UML диаграмма объектов

Диаграмма (см. рис. 2) представляет собой «снимок» системы во время процесса покупки билета пользователем через веб-сайт. В объекте webUser хранится информация о текущем пользователя: в данном случае важен лишь id пользователя в БД — и название текущей HTML-страницы.

В объекте v хранится такой же id как и у объекта webUser (в данном случае роль пользователя — посетитель) и список купленных билетов.

В объекте билета t имеется идентификатор билета в БД, id владельца и его цена.

pe — объект, соответствующий записи в таблице БД с информацией о проведенных транзакциях. Хранящаяся информация: заплаченная сумма и id покупателя.

В объекте pt представляет собой экземпляр проведенной транзакции, в хранится заплаченная сумма.

3.2.3. UML диаграмма компонентов

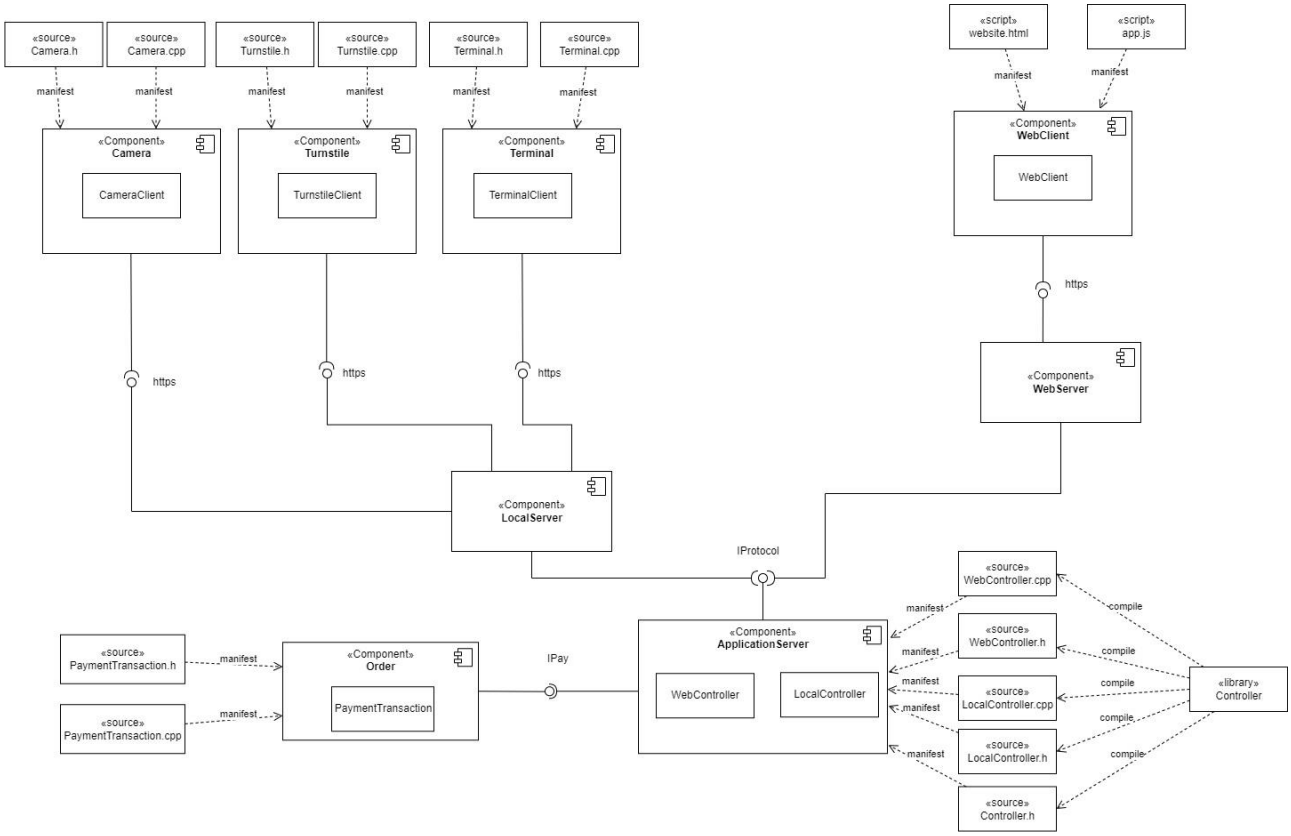

Рисунок 14 – UML диаграмма компонентов

На данной диаграмме показаны особенности физического представления системы. Компонент WebClient, представляющий клиентскую сторону веб-приложения, общается с компонентом WebServer, который возвращает клиенту ответы в виде html-страниц. Компоненты Camera, Turnstile, Terminal объединяются в одну локальную сеть с LocalServer. LocalServer и WebServer взаимодействуют с ApplicationServer для обработки запросов. Для взаимодействия с платежными системами ApplicationServer использует компонент Order.

3.2.4. UML диаграмма пакетов

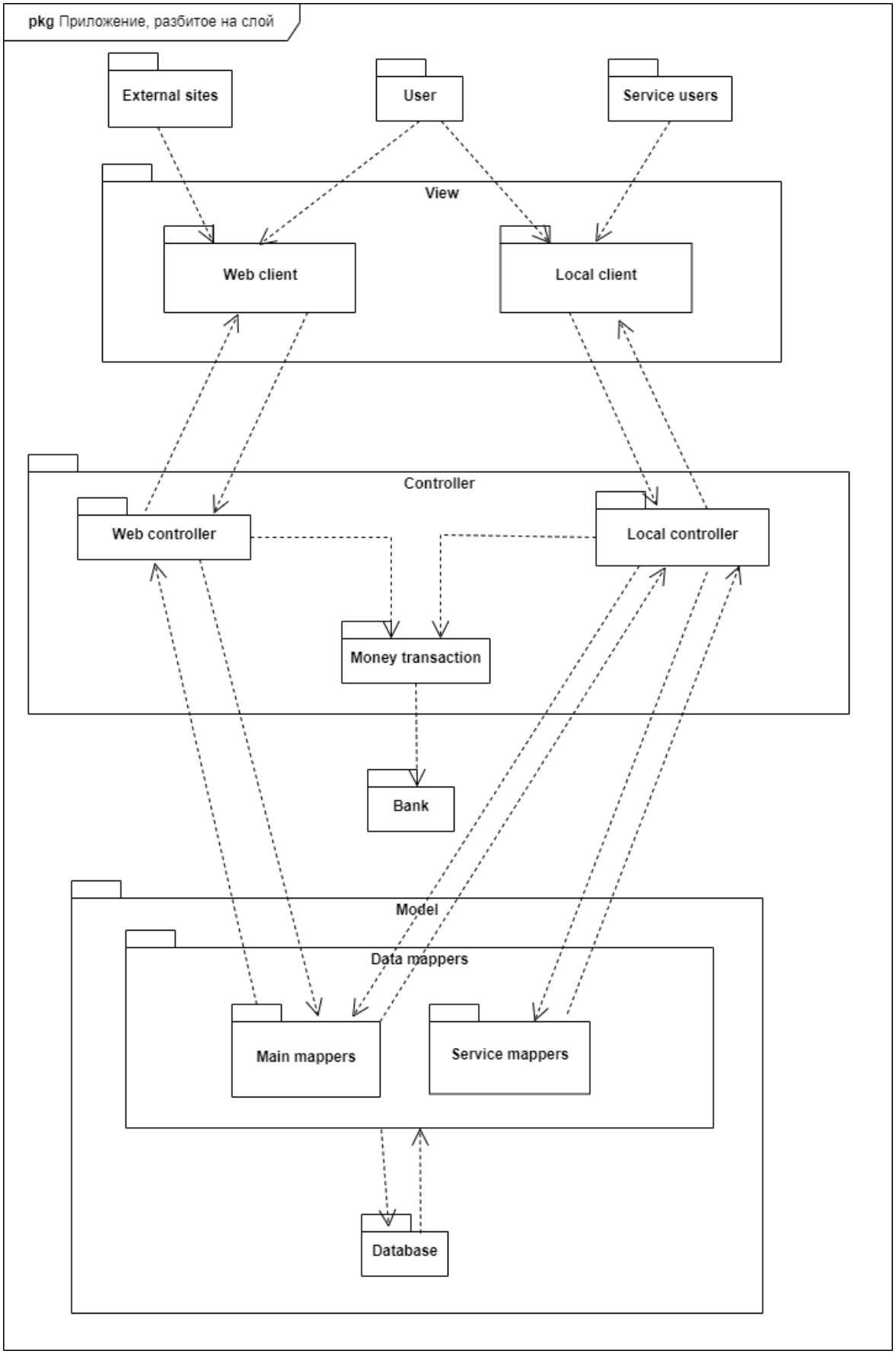

Рисунок 15 – UML диаграмма пакетов

Вся система разбита на слой: Model, View и Controller.

External sites — сторонние сайты для покупки билетов, которые взаимодействуют с разрабатываемой системой с помощью пакета Web client. User — пользователь, который зашел на сайт стадиона для покупки билета. Service users — служебные пользователи: терминалы, камеры и турникеты, которые взаимодействуют с контроллером с помощью локального клиента.

Web controller и Local controller содержат контроллеры для взаимодействия по Интернету и по локальной сети соответственно. Money transaction содержит модули для выполнения банковских транзакции. Bank — банк, с которым происходит взаимодействие во время транзакции.

Data mappers — пакет, содержащий модули для «маппинга» записей из БД в экземпляры соответствующих классов. Main mappers - «мапперы», которыми пользуются обычные пользователи, Service mappers — служебные «мапперы», с которыми в основном работают служебные устройства: турникеты, камеры и терминалы. Database — пакет, содержащий модули для работы с БД.

3.3. Решения по средствам и способам связи для информационного обмена между компонентами системы

3.3.1. UML диаграмма последовательности процесса авторизации пользователя.

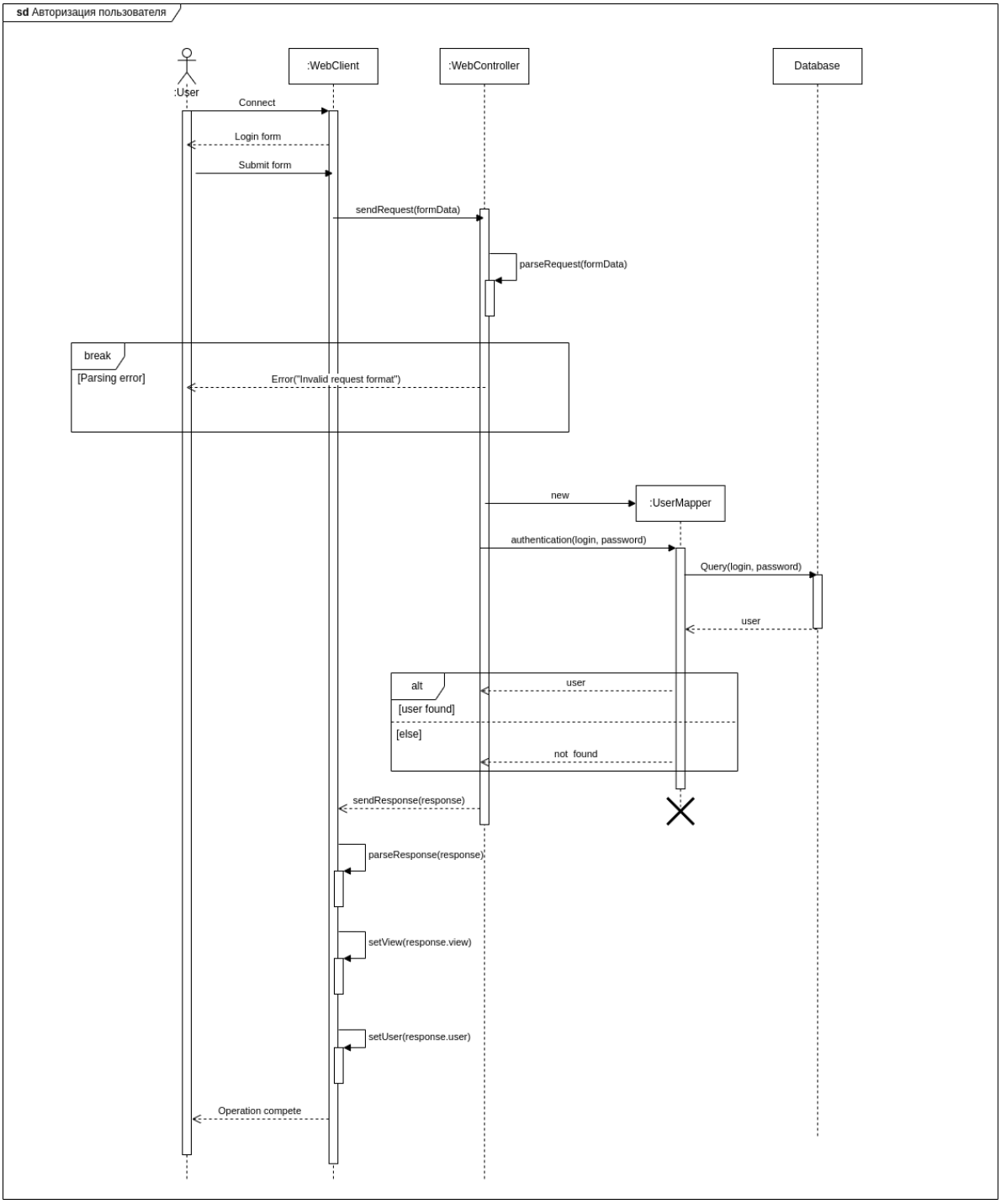

Рисунок 16 – UML диаграмма последовательности процесса авторизации пользователя

Диаграмма последовательности, описывающая процесс авторизации пользователя. Сначала актор (User) подключается в веб-клиенту (WebClient) и заполняет форму для авторизации. Далее веб-клиент формирует запрос и отправляет его на веб-сервер приложения (WebController), где запрос парсится. В случае, если парсинг данных не удался, пользователю возвращается страница с информацией об ошибке. Затем контроллер создает экземпляр объекта UserMapper для поиска пользователя в БД по введенным логину и паролю. Если соответствующей записи нет в БД, то пользователю возвращается информация об этом. В ином случае, на веб-клиент возвращается ответ от сервера, который парсится. Затем с помощью сеттеров обновляются соответствующие поля веб-клиента, и процесс авторизации завершен.

3.3.2. UML диаграмма последовательности (Sequence Diagram) процесса покупки билетов через сайт.

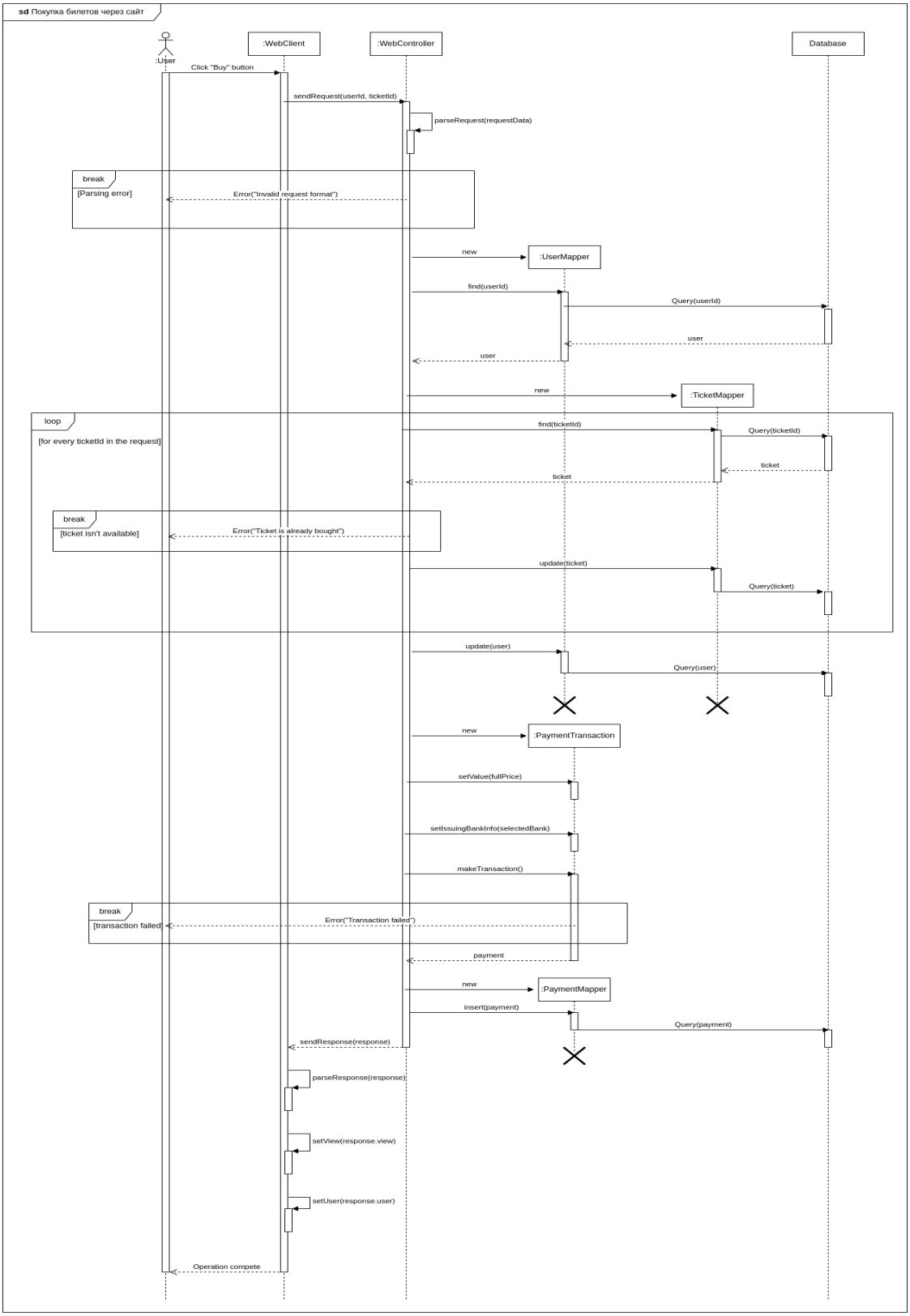

Рисунок 17 – UML диаграмма последовательности процесса покупки билетов через сайт

Диаграмма последовательности, описывающая процесс покупки билета пользователем через веб-сайт. Сначала актор (User) нажимает на кнопку «Купить», затем веб-клиент формирует запрос и отправляет его на веб-сервер приложения (WebController), где запрос парсится. В случае, если парсинг данных не удался, пользователю возвращается страница с информацией об ошибке. Затем контроллер создает экземпляр объекта UserMapper для поиска пользователя, который отправил запрос. Далее контроллер создает экземпляр объекта TicketMapper для поиска выбранных пользователем билетов в БД. Если хотя бы один из выбранных билетов уже куплен, то пользователю возвращается ошибка с информацей об этом. В ином случае в БД обновляется информация о каждом выбранном пользователем билете (записываются данные о покупателе). Затем обновляется информация о самом пользователе, который выполнил запрос (обновляется список купленных билетов). Затем создается инстанс класса PaymentTransaction для проведения банковской транзакции, устанавливаются цена билета, выбранный пользователем банк-эмиттер и выполняется сама транзакция. Если во время транзакции произошла ошибка, то пользователю выводится сообщение об этом. Иначе в БД с помощью класса PaymentMapper заносятся данные о новой транзакции. Далее информация на стороне пользователя обновляется и выводится сообщение об успешном завершении операции.

3.3.3. UML диаграмма последовательности процесса покупки билетов через терминал.

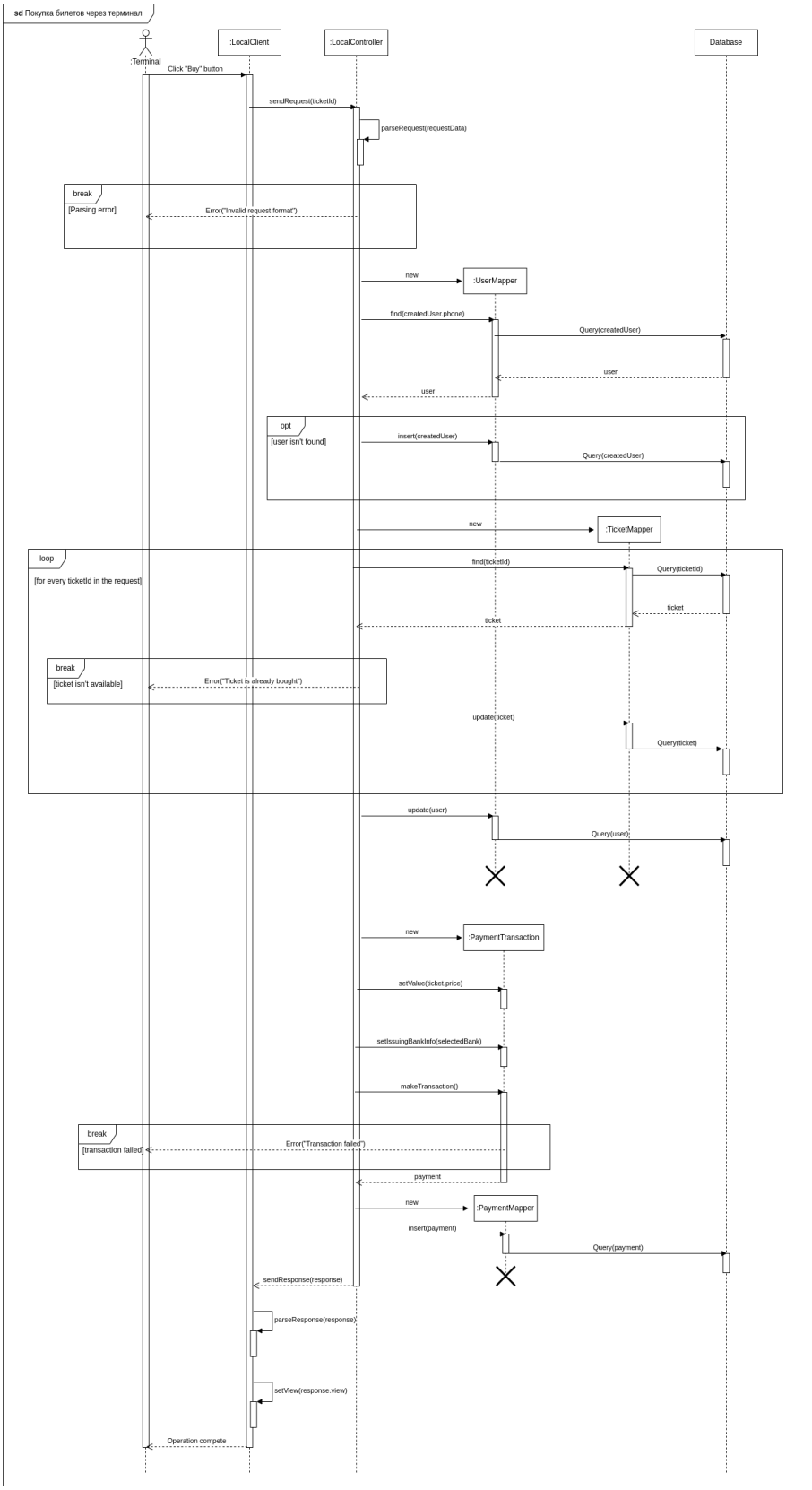

Рисунок 18 – UML диаграмма последовательности процесса покупки билетов через терминал

Процесс покупки билета через терминал (см. рис. 13) отличается от покупки билета через веб-сайт тем, что если пользователь впервые покупает билеты, то предоставленные им данные (например, телефон) заносятся в БД. В ином случае, последовательность действий аналогична тем, которые представлены на предыдущей диаграмме (см. рис. 12).

3.3.4. Диаграмма коммуникации

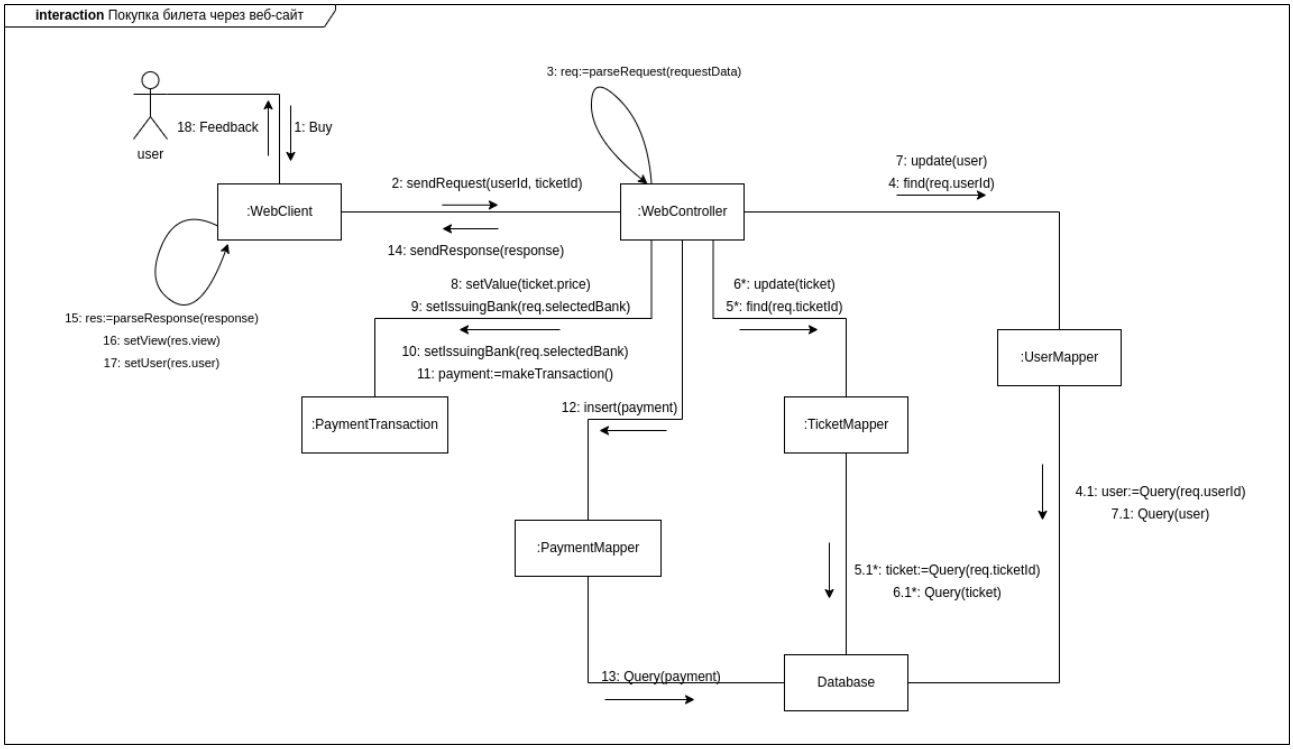

Рисунок 19 – UML диаграмма коммуникации

Семантически предоставленная диаграмма (см. рис. 14) похожа на диаграмму последовательности для покупки билета через веб-сайт (см. рис. 12). Здесь лишь добавлены связи между экземплярами классов.

3.4. Решения по входным и выходным документам и сообщениям, последовательности обработки информации, алгоритмам процедур и операции

3.4.1. UML диаграмма состояний мероприятия

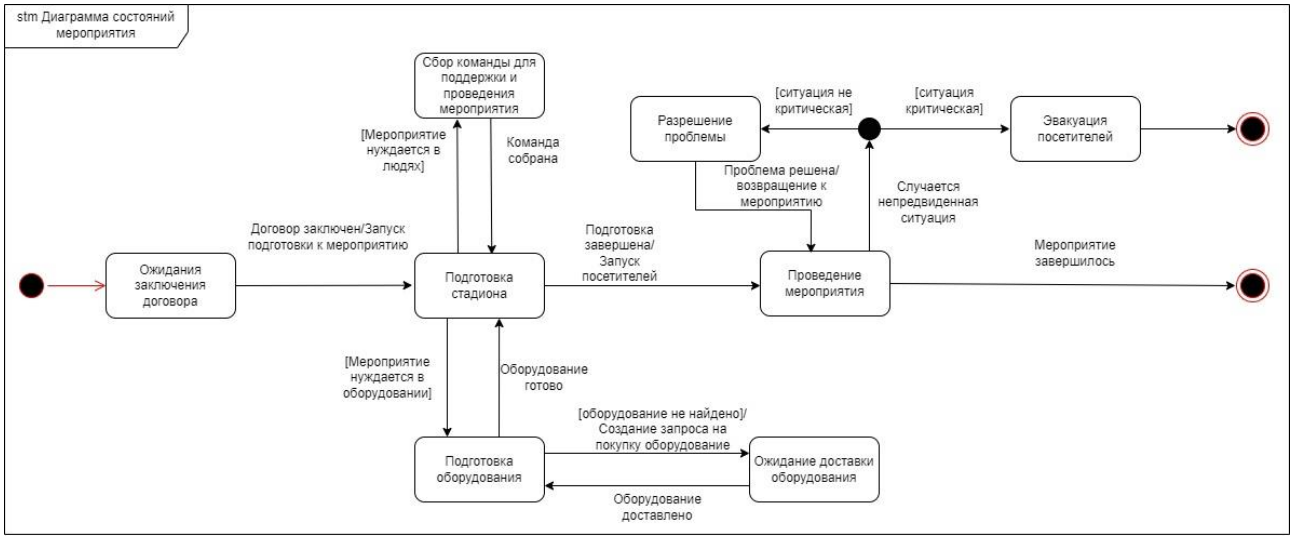

Рисунок 20 – UML диаграмма состояний для мероприятия

На данной диаграмме представлены состояния мероприятия. Изначально система находится в режиме ожидания, пока арендатор не закажет мероприятие и не подпишет договор с администрацией. Далее происходит подготовка стадиона. В данном состоянии подготавливают стадион к надлежащему виду (например, если нужно подстричь газон) и выделяют места на стадионе в соответствии с договором. Если мероприятию необходимо оборудование от стадиона, оно устанавливается в состоянии “Подготовка оборудования” и при необходимости закупается. Так же мероприятию может понадобиться команда поддержки для работы с посетителями, тогда соответствующие люди ищутся в состоянии “Сбор команды для поддержки и проведения мероприятия”. Когда подготовка будет завершена, мероприятие перейдёт в состояние “Проведение мероприятия”, в котором осуществляется контроль билетов и наблюдение за безопасностью посетителей. Если возникает внештатная ситуация, система оценивает уровень угрозы, выдаёт

соответствующие сообщение и персонал решает разобраться с проблемой на месте и продолжить мероприятие или эвакуировать посетителей и завершить.

3.4.2. UML диаграмма состояний покупки билетов

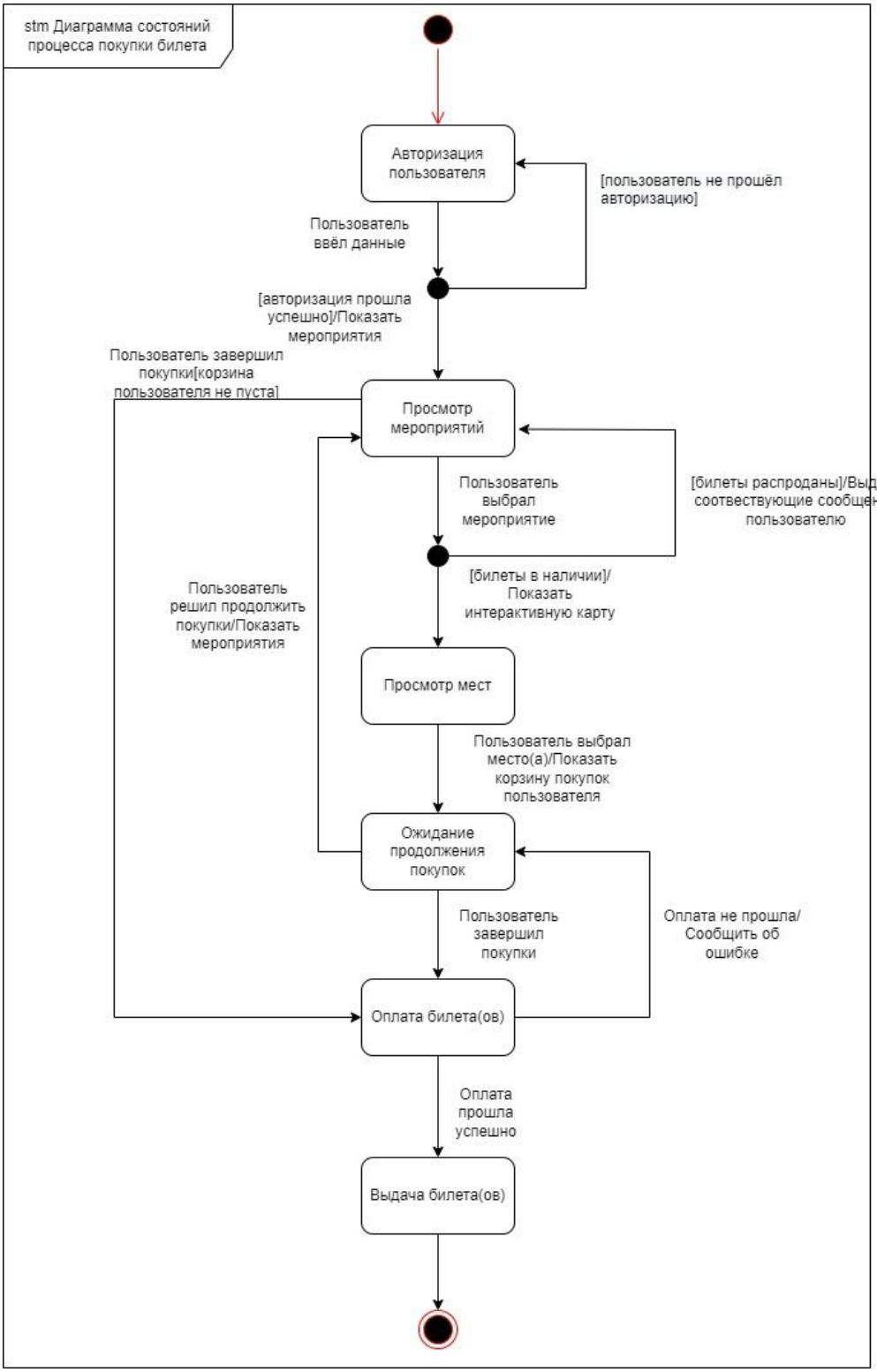

Рисунок 21 – UML диаграмма состояний для процесса покупки билета

На данной диаграмме представлены состояния процесса покупки билета. Изначально пользователю необходимо авторизоваться в системе (терминалы уже авторизированы в локальной сети системы). В

данном состоянии проверяется логин и пароль пользователя. Далее система в состоянии “просмотр мероприятий” выдаёт список ближайших мероприятий на стадионе. Если пользователь нашёл интересующее его мероприятие и на него остались билеты, система переходит в следующее состояние “Просмотр мест”, в котором предлагает пользователю выбрать место на интерактивной карте. В следующем состоянии “Ожидание продолжения покупок” система кладёт выбранные билеты в виртуальную корзину и предлагает пользователю продолжить покупки (тогда процесс покупки вернётся к состоянию “Просмотр мероприятий”) или оплатить имеющиеся в корзине билеты. Далее в состоянии “Оплата билета(ов)” в зависимости от способа покупки (онлайн или у терминала) происходит процесс оплаты билета(ов). Если платёжным системам не удаётся совершить операцию, система возвращает пользователя в предыдущие состояние. В последнем состоянии “Выдача билета(ов)” система выдаёт пользователю купленные билеты.

3.4.3. UML диаграмма деятельности для процесса покупки билета

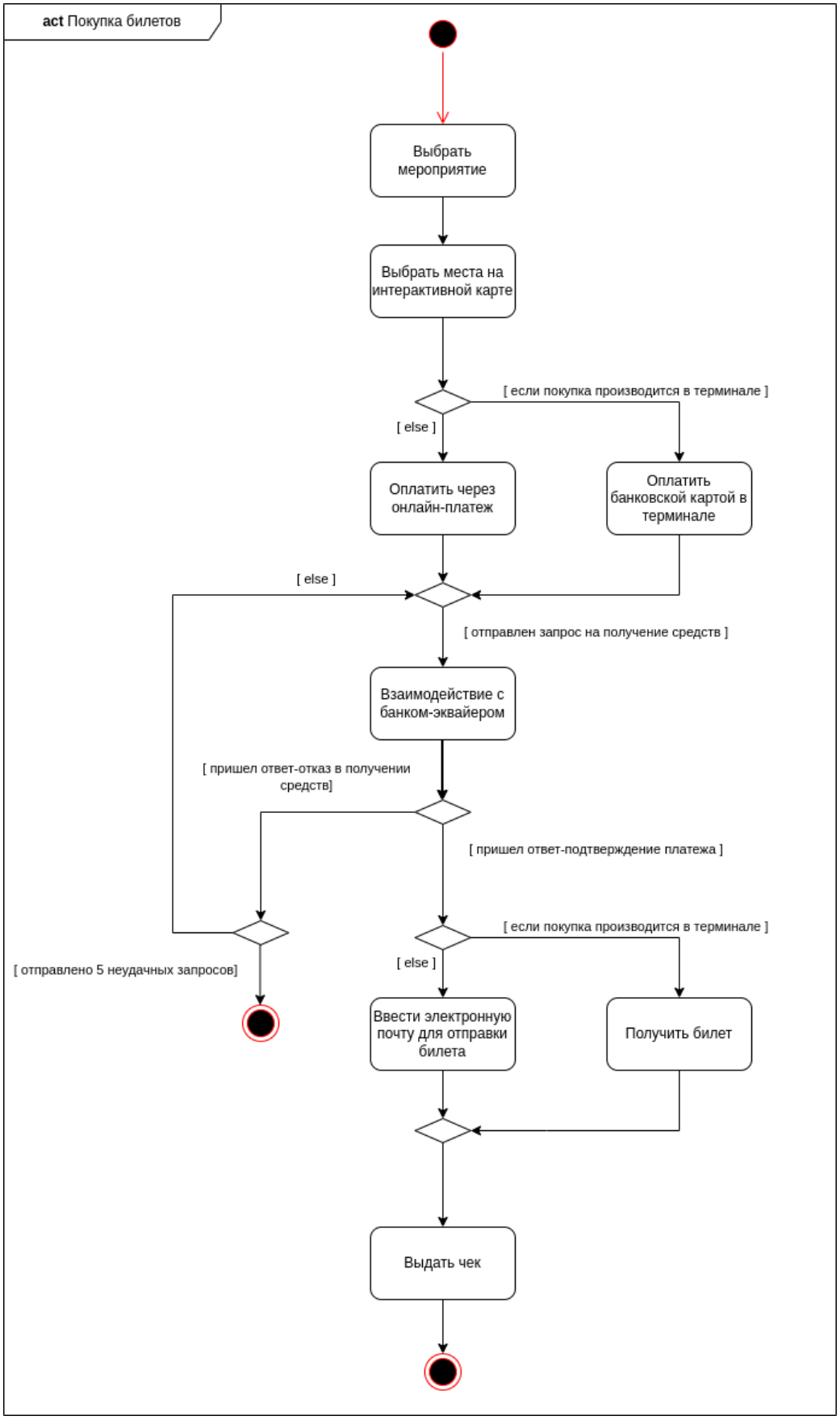

Рисунок 22 – UML диаграмма деятельности для процесса покупки билетов

«Выбрать мероприятие» - пользователь выбирает мероприятие, на которое хочет купить билет

«Выбрать места на интерактивной карте» - пользователь выбирает от 1 до 10 свободных мест на выбранном мероприятии

Если пользователь приобретает билет в терминале на стадионе, он должен «Оплатить банковской картой в терминале» стоимость билетов. Если же пользователь приобретает билет онлайн, то он должен «Оплатить через онлайн-платеж»

Далее идет взаимодействие с банковской системой: отправляется запрос в банк-эквайер на получение средств от клиента, в это время банк-эквайер налаживает связь с банком-эмитентом клиента, через который клиент производит оплату. После этого банк-эквайер возвращает ответ – результат операции. Ответ-подтверждение означает. Что операция прошла успешно, а ответ-отказ – что при оплате возникли проблемы. После 5 попыток на запрос средств покупка билетов останавливается.

Если оплата произведена успешно, то пользователь «Получает билет», если пользовался терминалом, или «Вводит электронную почту для отправки билета»

После этого пользователь «Получает чек»

3.4.4. UML диаграмма деятельности для аренды стадиона

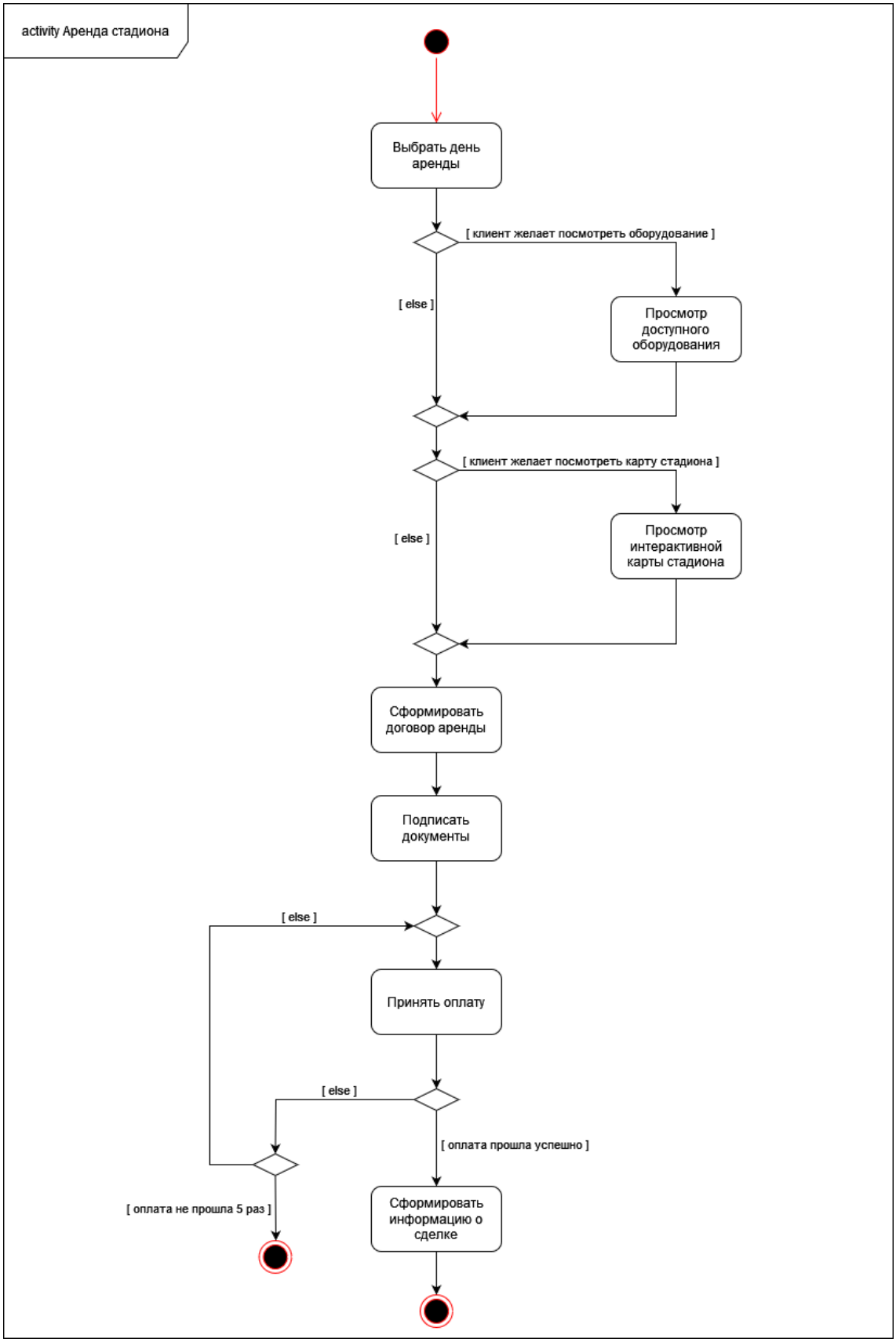

Рисунок 23 – UML диаграмма деятельности для аренды стадиона

«Выбрать день аренды» - пользователь выбирает день для проведения мероприятия

«Просмотр доступного оборудования» - пользователь может посмотреть все имеющееся на стадионе оборудование для проведения мероприятия

«Просмотр интерактивной карты стадиона» - пользователь может посмотреть карту стадиона для оценки количества мест и посадки

«Сформировать договор аренды» - подготовка необходимых документов для предоставления аренды

«Подписать документы» - окончательное оформление документов на аренду

Потом происходит процесс оплаты. «Принять оплату» - внесение суммы клиентом. Если при оплате возникли трудности, то даётся 5 попыток произвести оплату, после которых процесс аренды прерывается.

Если оплата прошла успешна, формируется и обрабатывается информация о сделке

3.5. Решения по комплексу технических средств, его размещению на объектах эксплуатации.

3.5.1. UML диаграмма развертывания.

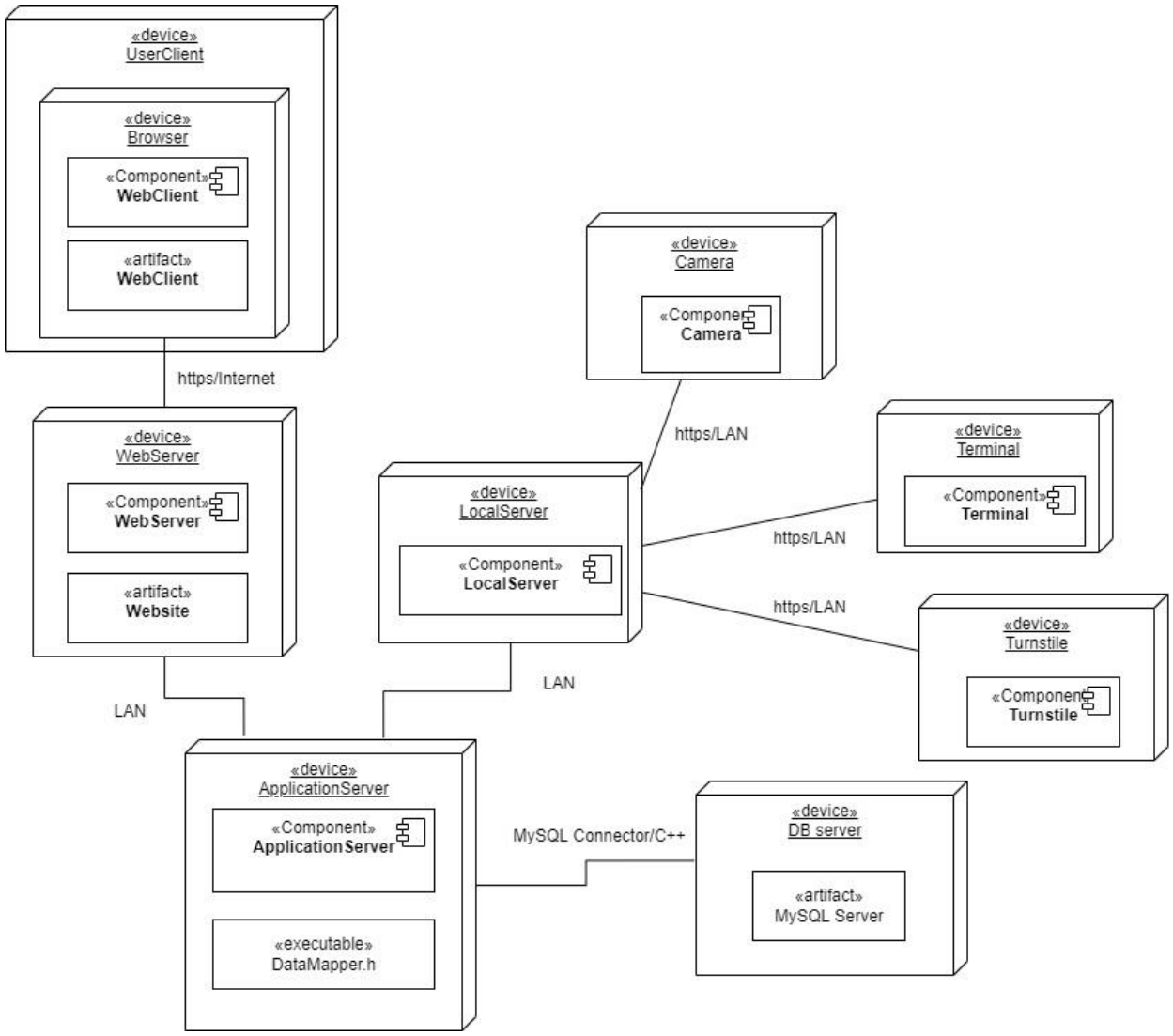

Рисунок 24 – UML диаграмма развертывания

На данной диаграмме (см. рис. 11) показаны как компоненты будут размещаться в физической структуре. UserClient по Internet соединяется с WebServer. WebServer, ApplicationServer, LocalServer, Camera, Terminal, Turnstile образуют единую закрытую локальную сеть. Обращение к базе данных MySQL происходит с помощью MySQL Connector/C++.

3.6. Решения по составу программных средств, языкам и технологиям реализации.

Система будет реализована с помощью языков программирования JavaScript (frontend) и C++ (backend).

Серверное программное обеспечение:

- Операционная система: Ubuntu Server 18.04.
- СУБД: Oracle MySQL 8.0.
- Веб-сервер: Nginx 1.21.4.

3.7. Перечень и описание шаблонов проектирование, использованных при проектировании архитектуры.

3.7.1. Шаблон проектирования для «расслоения» архитектуры системы

При проектировании архитектуры системы был использован шаблон проектирования MVC (Model-View-Controller).

Слой View отвечает за взаимодействие с пользователем, определяет внешний вид соответствующих компонентов и способы его использования. В качестве слоя View выступают следующие классы: Client, WebClient, LocalClient, TerminalClient, TurnstileClient, CameraClient.

Слой Model ответственен за данные и методы работы с ними, он просто предоставляет доступ к данным и управлению ими, не завися от представления (View) и контроллера (Controller). Классы, принадлежащие данному слою, представлены на рисунке ниже.

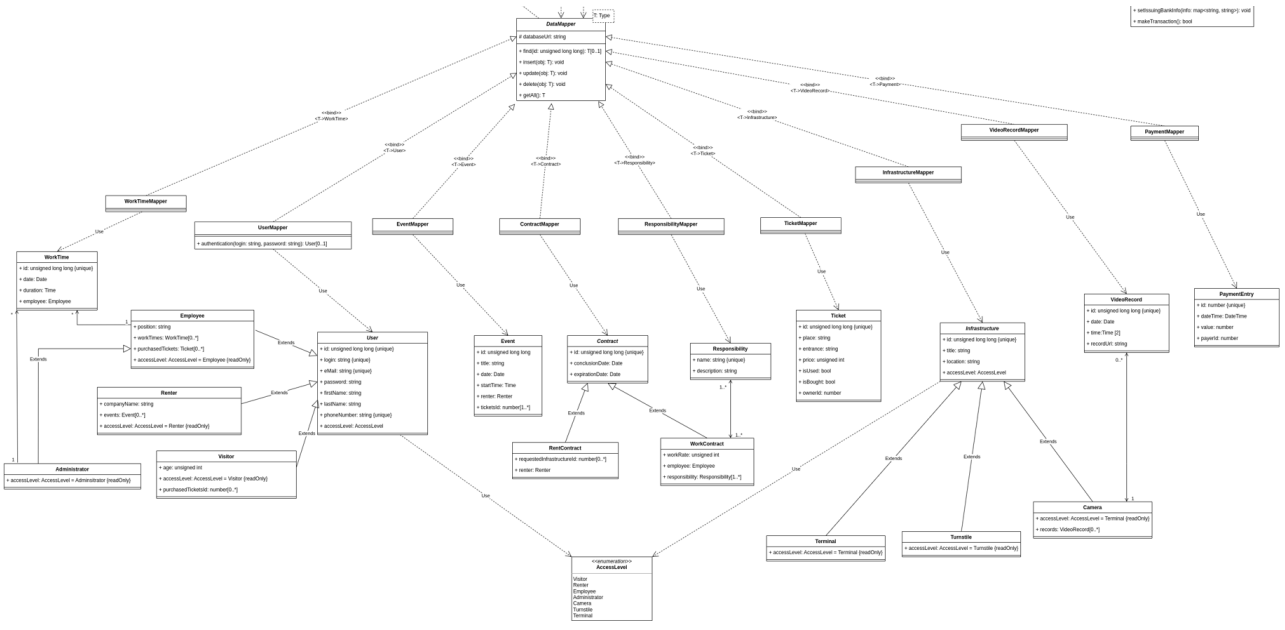

Рисунок 25 – Классы, принадлежащие слою Model

Слой Controller призван обеспечивать «связь» между пользователем и системой. Он направляет данные от пользователя к системе и наоборот. Также он ответственен за взаимодействие с

банками при оплате/возврате билетов данному слою принадлежат следующие классы: Controller, WebController, LocalController, PaymentTransaction.

Шаблон MVC использован для повышения повторного использования кода, который будет написан при реализации системы, т. е. чтобы в случае необходимости, можно было бы изменить View, не меняя контроллер, и наоборот.

3.7.2. Шаблоны проектирования для представления бизнес-логики и работы с данными.

Для проектирования бизнес-логики системы был выбран шаблон проектирования “Модель предметной области”

Типовое решение модель предметной области предусматривает создание сети взаимосвязанных объектов, каждый из которых представляет некую осмысленную сущность, содержащую как поведение (функции), так и свойства (данные). Реализация модели предметной области означает пополнение приложения целым слоем объектов, описывающих различные стороны предметной области.

В данной системе используется «простая» разновидность шаблона «Модель предметной области», так как одному объекту домена соответствует одна реляционная таблица. Использование данного шаблона позволяет упростить понимание диаграммы классов, причем «Модель предметной области» напоминает соответствующую базу данных. В качестве примера можно привести класс «Emploee» (Работник), который соответствует реляционной таблице, в которой имеются атрибуты id, login, password, eMail, position, workTimes и т. д.

В приведенной системе «поведение» инкапсулируется не в сами классы, относящиеся к предметной области, а в классы-преобразователи (mapper).

Шаблон «Data mapper» позволяет упростить взаимодействие с БД и сделать его более гибким. Данный шаблон хорошо работает в связке с паттерном «Domain model». Рассмотрим, например, класс «Event» (Событие). Контроллер при необходимости создает соответствующий класс, в данном случае это EventMapper, который инкапсулирует в себе код обращения к БД и выполнения определенных операции (поиск, вставка, обновление, удаление). То есть классы модели предметной области хранят только данные и не знают о существовании БД. Для каждого события выпускается несколько билетов, в БД это будет отражаться в виде внешних ключей в реляционной таблице «Событие», который ссылаются на ключи из таблицы «Билет». В задачи «преобразователя» (mapper) будет входить нахождение всех билетов, на которые ссылается определенное событие, и предоставление этой информации в виде поля-массива (в данном примере это ticketsId).

Участок диаграммы классов, демонстрирующий данные шаблоны проектирования, приведен на рис. 25.

ЗАКЛЮЧЕНИЕ.

В данном разделе представлены сведения об выполнении заданных в техническом задании требовании.

Требования	Реализация
4.1.1.1. Перечень подсистем, их назначение, основные характеристики, требования к числу уровней иерархии и степени централизации системы	Диаграмма классов (стр. 25-27) и ER-модель (стр. 18-20)
4.1.1.2. Требования к способам и средствам связи для информационного обмена между компонентами системы	Диаграмма развертывания (стр. 48)
4.1.1.3. Требования к характеристикам взаимосвязей создаваемой системы со смежными системами, требования к ее совместимости	Диаграмма IDEF0 2-го уровня, детализирующая процесс покупки и возврата билета (A2) (стр. 15). Класс PaymentTransaction на диаграмме классов (стр. 25-27)
4.1.1.4. Требования к режимам функционирования системы	Диаграмма развертывания (стр. 48)
4.1.1.5. Требования по диагностированию системы	Класс LogMessage на диаграмме классов (стр. 25-27) предназначен для хранения логов работы системных компонентов
4.1.1.6. Перспективы развития, модернизации системы	В диаграмме классов (стр. 25-27) используются шаблоны программирования для достижения гибкости и масштабируемости системы
4.1.2. Требования к численности и квалификации персонала системы и режиму его работы	В диаграмме классов (стр. 25-27) имеется перечисление AccessLevel (Administrator, DevOps, ContentManager)
4.1.3.1. Степень приспособляемости системы к изменению процессов и методов управления к отклонению параметров объекта управления	Класс LogMessage на диаграмме классов (стр. 25-27) позволяет отслеживать и просматривать отклонения от нормы в работе системных компонентов
4.1.3.2. Допустимые пределы модернизации и развития системы	Класс LogMessage на диаграмме классов (стр. 25-27) позволяет отслеживать скорость реакции системы при сбоях
4.1.4. Требования к надежности	Проверка требовании была произведена расчетным путем.
4.1.5. Требования к безопасности	Выполнение данного требования производится на аппаратном уровне

4.1.6. Требования к эргономике и технической эстетике	Данное требование выполняется в классах WebClient и TerminalClient на диаграмме классов (стр. 25-27) и на диаграмме последовательности покупки билета через сайт и терминал (стр. 35-38)
4.1.7. Требования к транспортабельности для подвижных АС	Требование не предъявляется.
4.1.8. Требования к эксплуатации, техническому обслуживанию, ремонту и хранению компонентов системы	Выполнение данного требования отслеживается на этапе развертывания аппаратных компонентов системы и её эксплуатации.
4.1.9. Требования к защите информации от несанкционированного доступа	Проверка уровня доступа производится в Controller на диаграмме классов (стр. 25-27). Процесс авторизации описан на диаграмме IDEF0 3-го уровня, детализирующей процесс авторизации пользователя (A11) (стр. 14). Авторизация пользователя в системе также описана на Use case диаграмме (стр. 23).
4.1.10. Требования по сохранности информации при авариях	Реализовано в объекте базы данных на диаграмме классов (стр. 25-27)
4.1.11. Требования к средствам защиты от влияния внешних воздействий	Выполнение данного требования отслеживается на аппаратном уровне
4.2. Требования к функциям (задачам), выполняемым системой (подсистема «Покупатель»)	Диаграмма IDEF0 2-го уровня, детализирующая процесс обработки на стороне сайта стадиона (A1) (стр. 12), диаграмма IDEF0 2-го уровня, детализирующая процесс покупки и возврата билета (A2) (стр. 15), Use case диаграммы подсистем «Оплата», «Пользователь» и «Взаимодействие с системой при посещении стадиона» (стр. 21, 23, 24), диаграмма классов (стр. 25-27), UML диаграмма деятельности для процесса покупки билета (стр. 44-45), UML диаграмма состояний покупки билетов (стр. 42-43), UML диаграмма последовательности процесса покупки билетов через сайт и терминал (стр. 35-38)
4.2. Требования к функциям (задачам), выполняемым системой (подсистема «Арендатор»)	Диаграмма IDEF0 2-го уровня, детализирующая процесс обработки на стороне сайта стадиона (A1) (стр. 12), Use case диаграммы подсистем "Аренда", "Пользователь" и "Взаимодействие с системой при посещении стадиона" (стр. 22, 23, 24), диаграмма классов (стр. 25-27), UML диаграмма деятельности для аренды стадиона (стр. 46-47).

4.2. Требования к функциям (задачам), выполняемым системой (подсистема «Сотрудник»)	Диаграмма IDEF0 2-го уровня, детализирующая процесс обработки на стороне сайта стадиона (A1) (стр. 12), Use case диаграммы подсистем «Оплата», «Пользователь» и «Взаимодействие с системой при посещении стадиона» (стр. 21, 23, 24), диаграмма классов (стр. 25-27).
4.2. Требования к функциям (задачам), выполняемым системой (подсистема «СКУД»)	Диаграмма классов (стр. 25-27), диаграмма IDEF0 1-го уровня (A0) (стр. 11), ER-модель (стр. 18-20).
4.3.1. Требования к математическому обеспечению	Требования не предъявляются
4.3.2. Требования к информационному обеспечению	Диаграмма компонентов (стр. 30), диаграмма развертывания (стр. 48) и диаграмма классов (стр. 25-27).
4.3.3. Требования к лингвистическому обеспечению	Классы WebClient и TerminalClient в диаграмме классов (стр. 25-27)
4.3.4. Требования к программному обеспечению	Требование не предъявляются.
4.3.5. Требования к техническому обеспечению	Выполнение данного требования отслеживается на этапе развертывания.
4.3.6. Требования к метрологическому обеспечению	Требование не предъявляются.
4.3.7. Требования к организационному обеспечению	AccessLevel, Employee, Controller в диаграмме классов (стр. 25-27)
4.3.8. Требования к методическому обеспечению	Требование не предъявляются.
4.3.9. Требования к другим видам обеспечения системы	Требование не предъявляются.

Цели	**Реализация**
Уменьшение времени реакции на чрезвычайные ситуации в 1.5 – 1.7 раза	Во время подозрительной активности, зафиксированной камерой, происходит автоматической оповещение охраны сектора с угрозой (метод alarm в классе CameraClient в диаграмме классов (стр. 25-27), камера знает свое местонахождение через поле location). Также происходит автоматическое блокирование турникетов в секторе с подозрительной активностью (LocalController вызывает метод block у

	турникета (класс TurnstileClient в диаграмме классов (стр. 25-27)), местонахождение блокируемого турникета определяется в методе LocalController.alarm). Обработка ЧП во время мероприятия описана на диаграмме состояния мероприятия (стр. 40-41).
Уменьшение количества чрезвычайных ситуаций на 30%	С помощью камер отслеживается подозрительная активность. Охранники смотрят на камеру и реагируют в случае ЧП.
Уменьшение количества посредников при передаче указаний между персоналом и командой управления	С помощью сайта администрация может общаться с сотрудниками через личный кабинет
Добавление новых способов покупки билетов	Покупатели могут приобрести билеты через терминал или через сайт.
Добавление дистанционного взаимодействия между Заказчиком и арендаторами	Арендатору необходимо лично заключить договор о сотрудничестве. В договоре имеются данные для входа на сайт с правами арендатора. Через сайт арендатор может дистанционно организовать мероприятие.
Формирование любой отчётности не должно требовать ручных операций.	Автоматическая генерация договоров о трудоустройстве, о сотрудничестве, квитанции о покупках.