**Министерство образования и науки Забайкальского края**

**Государственное профессиональное образовательное учреждение**

**«ЧИТИНСКИЙ ТЕХНИКУМ ОТРАСЛЕВЫХ ТЕХНОЛОГИЙ И БИЗНЕСА»**

**(ГПОУ «ЧТОТиБ»)**

**КУРСОВОЙ ПРОЕКТ**

по дисциплине «Технология разработки программного обеспечения»

«АВТОМАТИЗИРОВАННОЕ РАБОЧЕЕ МЕСТО «ГОСТИНИЦА»»

Выполнил:  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ М.Э. Рахимов,

студент группы ИСиП-20-2п

«\_\_\_» \_\_\_\_\_\_\_\_\_\_2022 г.

Руководитель работы:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Д.А. Ломов

«\_\_\_» \_\_\_\_\_\_\_\_\_2022 г.

2022 г.

Оглавление

[Введение 3](#_Toc103327151)

[1. Теоретическая часть 4](#_Toc103327152)

[2. Практическая часть 11](#_Toc103327153)

[2.1. Диаграмма Вариантов Использования 11](#_Toc103327154)

[2.2. Диаграмма Последовательности 12](#_Toc103327155)

[2.3. Диаграмма Коммуникации 13](#_Toc103327156)

[2.4. Диаграмма Развертывания 13](#_Toc103327157)

[2.5. Диаграмма Деятельности 14](#_Toc103327158)

[2.6. Диаграмма Состояний 15](#_Toc103327159)

[2.7. Диаграмма Классов 15](#_Toc103327160)

[2.8. Диаграмма Компонентов 16](#_Toc103327161)

[Заключение 18](#_Toc103327162)

[Список использованной литературы 19](#_Toc103327163)

[Приложения 20](#_Toc103327164)

**Введение**

Унифицированный язык моделирования (называемый UML) был разработан с целью обеспечения единого визуального языка с богатейшей семантикой и развернутым синтаксисом для проектирования и внедрения различных программных систем со сложной структурой и поведением. UML представляет собой некий стандарт, используемый во многих отраслях, например, схематизации потоков производственных процессов. UML поддерживает диаграммы нескольких типов.

Диаграммы UML описывают границы, структуру и поведение как всей системы, так и отдельных объектов. Следует перечислить некоторые из них: классов, компонентов, развертывания, объектов, активности, коммуникации, последовательности и другие.

UML обеспечивает своим пользователям удобную и последовательную методологию и представляет собой набор эффективных методов схематизации и документирования различных аспектов моделирования программ и бизнес-систем.

Итак, целью курсового проекта является: закрепить знания, полученные на дисциплине «Технология разработки программного обеспечения», расширить и закрепить практические навыки построения диаграмм посредством построения диаграмм автоматизированного рабочего места «Гостиница».

Задачами курсового проекта являются:

* Закрепить опыт работы с программой Microsoft Visio и сервисом Visual Paradigm.
* Построить предложенные диаграммы c последующим описанием процесса.

Актуальность проекта заключается в следующем: с каждым годом туризм становится все популярнее, поэтому туристов, заселяющихся в гостиницы и отели становится больше, следствием этого является необходимость развивать автоматизированные места для более комфортной работы персонала с клиентами.

Объект исследования: предложенные для построения диаграммы UML.

Предмет исследования: унифицированный язык моделирования (UML).

Практическая значимость проекта: при расширении диаграмм и их интерпретации возможно применить проект для создания комфортного для работы автоматизированного рабочего места в гостинице.

1. **Теоретическая часть**
   1. **Диаграмма Вариантов Использования**

На диаграммах вариантов использования отображается взаимодействие между вариантами использования, представляющими функции системы, и действующими лицами, представляющими людей или системы, получающие или передающие информацию в данную систему. Из диаграмм вариантов использования можно получить довольно много информации о системе. Этот тип диаграмм описывает общую функциональность системы.

К базовым элементам диаграммы относятся: вариант использования и актер (иначе называя - действующее лицо).

На рисунке 1.1 продемонстрирован вариант использования (use case), который применяется для спецификации общих особенностей поведения системы или другой сущности без рассмотрения ее внутренней структуры (например, оформление заказа на покупку товара, получение информации о клиенте, отображение графической формы на экране монитора).

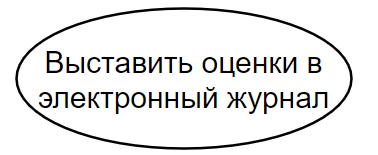


Рисунок 1.1 – Пример графической нотации варианта использования

Актер (actor) — это внешняя по отношению к моделируемой системе сущность, которая взаимодействует с системой и использует ее функциональные возможности для решения определенных задач. При этом актеры служат для обозначения согласованного множества ролей, которые могут играть пользователи в процессе взаимодействия с проектируемой системой. На рисунке 1.2 изображена графическая нотация актера в виде преподавателя.



Рисунок 1.2 – Пример графической нотации актера

Связи позволяют показать способы взаимодействия между элементами системы.

* 1. **Диаграмма Последовательности**

Диаграммы последовательностей являются частью диаграмм взаимодействия (используются для моделирования взаимодействий между объектами системы), более детального описания логики сценариев использования. Диаграмма последовательности позволяет изобразить поведение нескольких объектов в рамках одного прецедента, иначе называемая вариантом использования. Данная диаграмма является отличным средством для документирования проекта со стороны работы со сценариями. Диаграмма очень удобна для представления взаимодействия объектов внутри системы, но не для точного определения их поведения.

Диаграммы последовательностей обычно содержат объекты, которые взаимодействуют в рамках сценария, сообщения, которыми они обмениваются, и возвращаемые результаты, связанные с сообщениями. В общем, часто возвращаемые результаты обозначают лишь в том случае, если это не очевидно из контекста.

Объекты – элементы системы, с которыми происходит взаимодействие. Графическая нотация объекта изображена на рисунке 1.3.

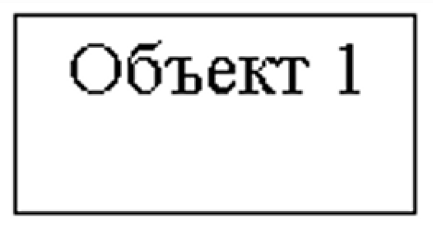


Рисунок 1.3 – Пример графической нотации объекта

Сообщения - элемент модели, предназначенный для представления отдельной коммуникации между линиями жизни некоторого взаимодействия. Таким образом, можно проследить связи на рисунке 1.4, обозначенные порядковыми номерами с наименованием необходимого действия.

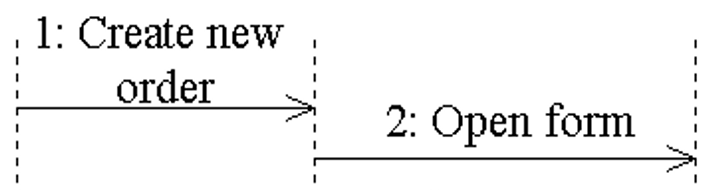


Рисунок 1.4 – Пример графической нотации сообщений

Прямоугольники на вертикальных линиях под каждым из объектов показывают «время жизни» (представляет одного индивидуального участника взаимодействия или отдельную взаимодействующую сущность) объектов. Впрочем, довольно часто их не изображают на диаграмме, все это зависит от индивидуального стиля проектирования.

* 1. **Диаграмма Коммуникации**

Диаграмма коммуникаций — это особый вид диаграмм взаимодействия, акцентированных на обмене данными между различными участниками взаимодействия. Вместо того чтобы рисовать каждого участника в виде линии жизни и показывать последовательность сообщений, располагая их по вертикали, как это делается в диаграммах последовательности, коммуникационные диаграммы допускают произвольное размещение участников, позволяя рисовать связи, показывающие отношения участников, и использовать нумерацию для представления последовательности сообщений.

* 1. **Диаграмма Развертывания**

Диаграмма развертывания – это тип UML-диаграммы, которая показывает архитектуру исполнения системы, включая такие узлы, как аппаратные или программные среды исполнения, а также промежуточное программное обеспечение, соединяющее их.

Диаграммы развертывания помогают моделировать аппаратную топологию системы по сравнению с другими типами UML-диаграмм, которые в основном описывают логические компоненты системы.

На рисунке 1.5 изображен узел (node), представленный в виде куба с заголовком, где прописывается наименование узла или компонента системы. Он представляет собой физическую сущность, которая выполняет одну или несколько компонентов, подсистем или исполняемых файлов. Узел может быть аппаратным или программным элементом системы.

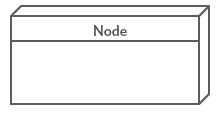


Рисунок 1.5 – Пример графической нотации узла

Артефакты (Artifact) – это конкретные элементы, которые вызваны процессом разработки. Примерами артефактов являются библиотеки, архивы, конфигурационные файлы, исполняемые файлы и другие возможные файлы. Артефакт изображается прямоугольником с изображением файла в правом верхнем углу. Пример графической нотации артефакта изображен на рисунке 1.6.



Рисунок 1.6 – Пример графической нотации артефакта

Таким образом следует сделать вывод, диаграммы развертывания используются для визуализации аппаратной части системы, каналов связи между объектами системы и размещения программных файлов на этом аппаратном обеспечении.

* 1. **Диаграмма Деятельности**

Диаграмма активности UML позволяет более детально визуализировать конкретный случай использования. Это поведенческая диаграмма, которая иллюстрирует поток деятельности через систему.

Диаграммы активности UML также могут быть использованы для отображения потока событий в бизнес-процессе. Они могут быть использованы для изучения бизнес-процессов с целью определения их потока и требований.

Для построения диаграммы Деятельности используют следующие символы:

* Начальный узел - используется для представления отправной точки или начального состояния деятельности. На рисунке 1.7 изображена его графическая нотация.



Рисунок 1.7 – Пример графической нотации начального узла

* Действие - используется для представления деятельности процесса. Обозначение действие на диаграмме продемонстрировано на рисунке 1.8.



Рисунок 1.8 – Пример графической нотации действия

* Конечный узел активности - используется для обозначения конца всех контрольных потоков в рамках деятельности. Данный элемент диаграммы изображен на рисунке 1.9.



Рисунок 1.9 – Пример графической нотации конечного узла

* Узел принятия решения - используется для представления условной точки ответвления с одним входом и несколькими выходами. При применении данного узла на практике, следует его изобразить в соответствии с рисунком 1.10.



Рисунок 1.10 – Пример графической нотации узла принятия решения

* Вилка - используется для представления потока, который может разветвляться на два и более параллельных потока. Графическая нотация изображена на рисунке 1.11.

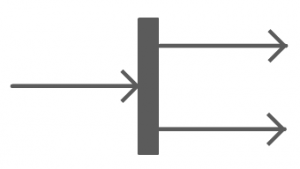


Рисунок 1.11 – Пример графической нотации вилки

Диаграмма деятельности отлично отражает последовательность действий и событий сценария. На ней можно допустить условия расширения действий. В прочем, данная диаграмма является очень полезной при проектировании проекта.

* 1. **Диаграмма Состояний**

Диаграмма состояний показывает, как объект переходит из одного состояния в другое. Диаграммы состояний служат для моделирования динамических аспектов системы. Данная диаграмма полезна при моделировании жизненного цикла объекта. От других диаграмм диаграмма состояний отличается тем, что описывает процесс изменения состояний только одного экземпляра определенного класса - одного объекта, причем объекта реактивного, то есть объекта, поведение которого характеризуется его реакцией на внешние события.

Диаграмма состояний содержит те же символы, что и диаграмма деятельности, но все узлы меняют название на узлы состояний. Таким образом, диаграмма деятельности является более развернутым вариантом диаграммы состояний.

* 1. **Диаграмма Классов**

Диаграмма классов определяет типы классов системы и статические связи, которые существуют между ними. На диаграммах классов изображаются также атрибуты классов, операции классов и ограничения, которые накладываются на связи между классами. Вид и интерпретация диаграммы классов существенно зависит от точки зрения (уровня абстракции): классы могут представлять сущности предметной области или элементы программной системы.

Основными элементами диаграммы являются классы и связи, среди которых можно выделить связь ассоциации, агрегации и наследования.

Класс представляет концепт, который описывает состояние (атрибуты) и поведение (методы). Пример графической нотации класса изображен на рисунке 1.12.

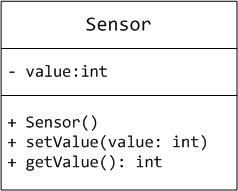


Рисунок 1.12 – Пример графической нотации класса

Диаграмма классов занимает центральное место в проектировании системы. Нотация классов используется на разных этапах проектирования продукта и строится с различной степенью детализации.

* 1. **Диаграмма Компонентов**

Диаграммы компонентов используются для визуализации организации компонентов системы и зависимостей между ними. Они позволяют получить высокоуровневое представление о компонентах системы.

Компонентами могут быть программные компоненты, такие как база данных или пользовательский интерфейс; или аппаратные компоненты, такие как схема, микросхема или устройство; или бизнес-подразделение, такое как поставщик, платежная ведомость или доставка.

В данном типе диаграмм используются следующие типы символов:

* Компонент – это основная составляющая диаграммы, позволяют определить компоненты системы, пример нотации которого продемонстрирован на рисунке 1.13.

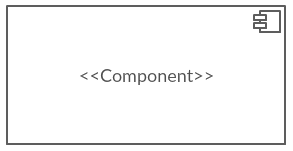


Рисунок 1.13 – Пример графической нотации компонента

* Интерфейсы - показывают, как компоненты соединены друг с другом и взаимодействуют друг с другом. Таким образом, чтобы изобразить интерфейс на диаграмме, необходимо иметь хоть один компонент, а между ними в виде круга с дугой (рисунок 1.14) предоставляют интерфейс.

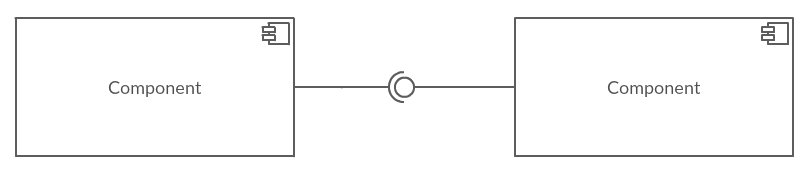


Рисунок 1.14 – Пример графической нотации интерфейсов

* Порты - используется, когда компонент делегирует интерфейсы внутреннему классу. Изображение порта на диаграмме продемонстрировано в виде квадрата на границе компонента на рисунке 1.15.

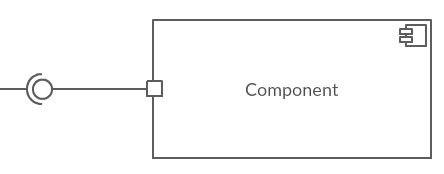


Рисунок 1.15 – Пример графической нотации порта

* Зависимости – показывают отношения компонентов в системе. Зависимость изображается на диаграмме в виде стрелки. Пример зависимости между компонентами изображен на рисунке 1.16.



Рисунок 1.16 – Пример графической нотации зависимости

В отличие от раннее рассмотренных диаграмм, диаграмма компонентов описывает особенности физического представления системы и позволяет определить архитектуру разрабатываемой системы.

1. **Практическая часть**
   1. **Диаграмма Вариантов Использования**

На диаграмме, продемонстрированной на рисунке 2.1, изображены три варианта использования (ВИ): ВИ администратор, ВИ event-менеджер и ВИ PR-менеджер.

Рассмотрим вариант использования администратора: стоит начать с того, что актером (или действующим лицом) является сам администратор, который может выполнять множество функций. Функции можно объединять в группы, чтобы было более удобно описывать связанные функции по логике в схеме. Первым блоком функций является бронирование, состоящее из функции забронировать (которую можно дополнить спальным местом и завтраком – это позволяет сделать связь extend) и функция отмены брони. Следующим блоком является заселение (check), которое в свою очередь включает в себя заселение (check-in) и выселение (check-out). Далее следует функция отслеживания статуса номера и произведение оплаты, которая может включать в себя (связь include) оплату наличными и картой.

Следующим вариантом использования является event-менеджер (менеджер по планированию и организации мероприятий в гостинице). В данном варианте использования есть актер и функции, связанные ассоциативной связью (определяет конкретную роль). В состав входят следующие функции: составление базы контактов, подсчет бюджета мероприятия, выбор помещения, составление договора с заказчиком и составление сметы и плана мероприятия.

Конечным вариантом использования является PR-менеджер (менеджер по продвижению гостиницы в сети). Этот вариант использования составляет актер (действующее лицо) – PR-менеджер и его функции. Функций у него немного, но все они являются важными для работы гостиницы. Первая функция – взаимодействие с общественностью, которая включает в себя (связь include) работу со Средствами Массовой Информации (СМИ) и социальными ведомствами. Второй функцией является оценка трафика гостей в заведении. И в заключении, пожалуй, самая важная его функция – это размещение публикаций в сети, состоящая из (связь include) размещения публикаций в ВК, Телеграм и на официальном сайте гостиницы.

Данные варианты использования являются базовыми для работы в гостинице, т.к. основные рабочие места у рабочих станций занимают лишь малая часть персонала. Поэтому таким образом, были задействованы три департамента гостиницы, в свою очередь, важные для функционирования и процветания гостиницы или отеля.

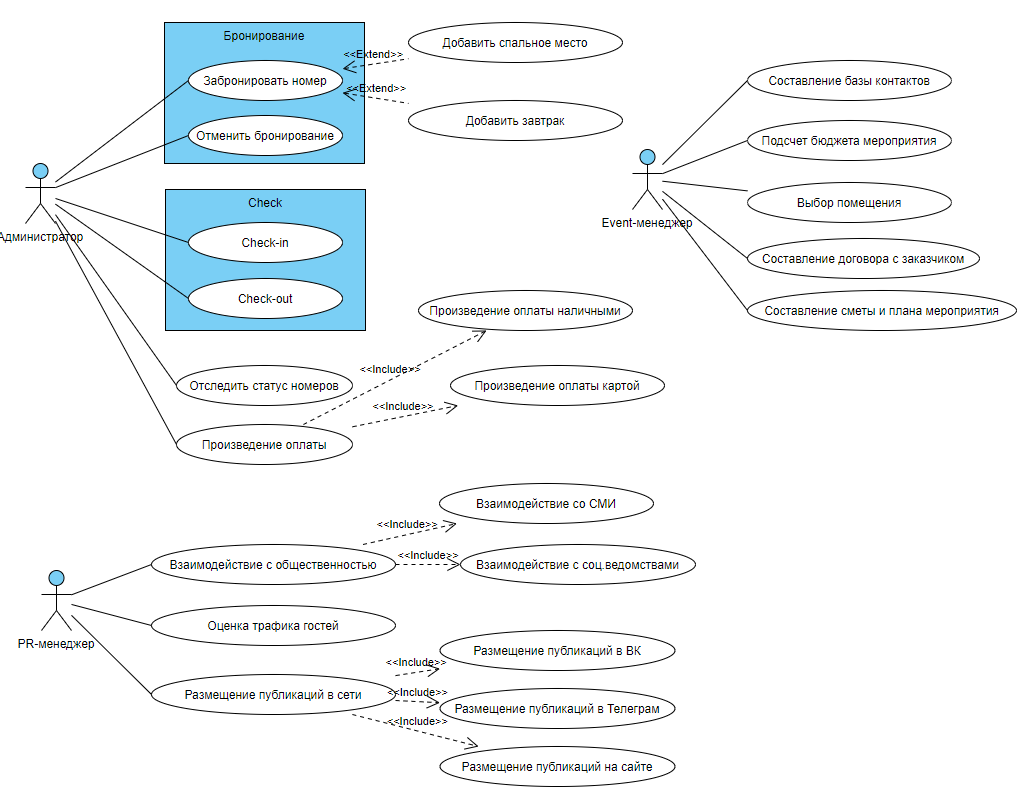


Рисунок 2.1 – Диаграмма Вариантов Использования

* 1. **Диаграмма Последовательности**

В диаграмме последовательности (Смотрите в Приложении А, рисунок А.1) описан процесс бронирования номера администратором. Актером (действующим лицом) является менеджер гостиницы или же отеля. И от линии жизни актера отходит первое сообщение – запрос на получение списка номеров на форму ListRoomsForm.cshtml. Через какое-то время будет отправлено второе сообщение на получение списка номеров к RoomsController.cs. Все эти прямоугольники в графической нотации будут обозначать объекты, между которыми происходит взаимодействие. Отсюда же будет отослано третье сообщение, являющееся запросом на получение данных из базы данных. Здесь линия сообщений закончится, и отправится назад, так как ответ на запрос был получен. Таким образом к контроллеру будут переданы полученные номера, а контроллер передаст форме этот список, в свою очередь форма отобразит этот список актеру и предложит выбрать.

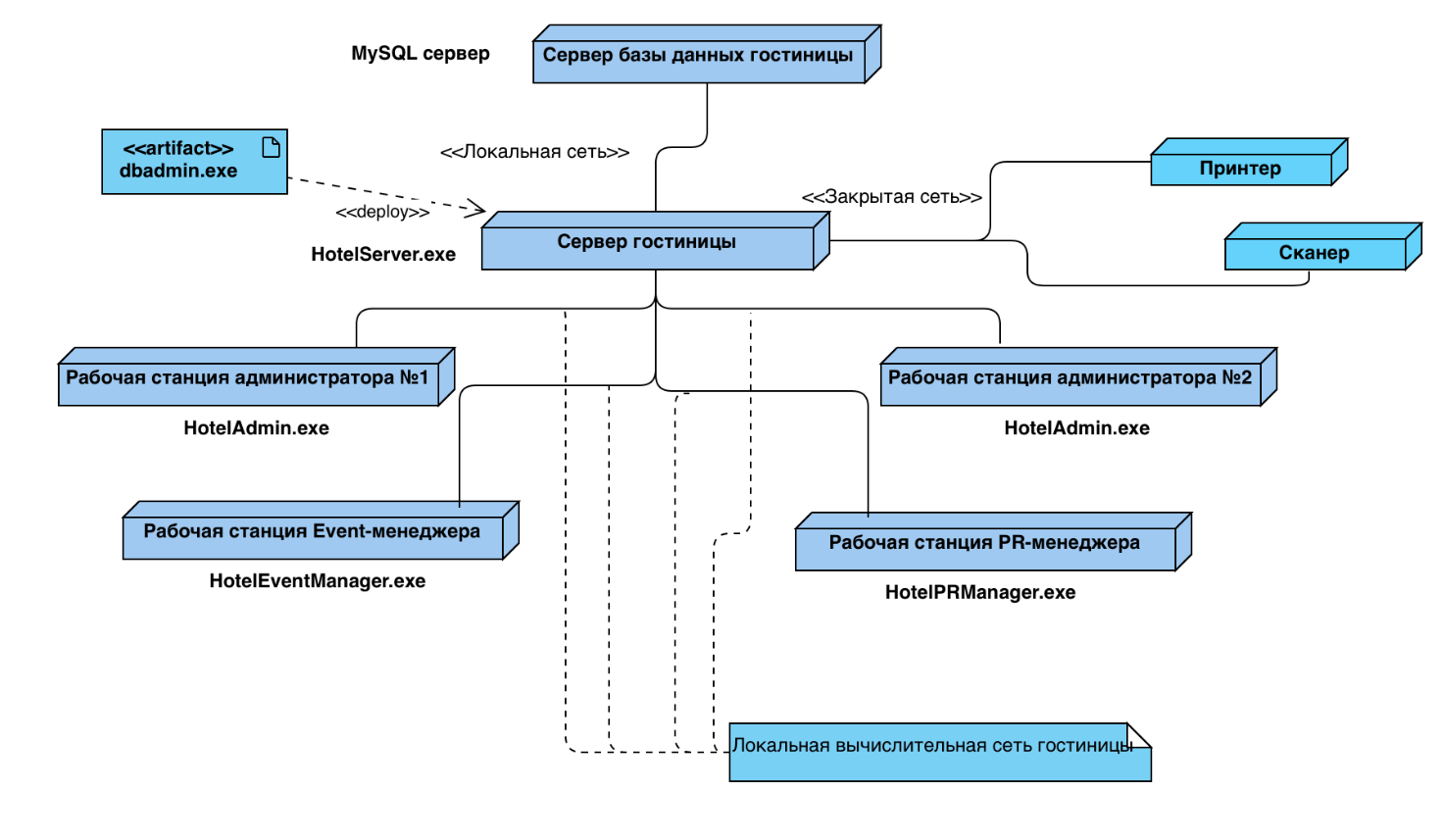
Спустя некоторое время актер выполнит выбор, а форма передаст сообщение контроллеру. Так получится, что RoomsController.cs получит данные о данном номере из базы данных и удостоверится, свободен ли номер через обращение к самому себе (на диаграмме стрелка возвращается в этот же период жизни элемента). Через некоторый ряд сообщений актер узнает статус и запустит процесс бронирования, обращаясь уже к форме BookingForm.cshtml. Здесь мы можем увидеть ветвление – если введенные данные корректны, то будет отправлено сообщение с результатом (пунктирная линия) к актеру «Продолжение бронирования», а иначе «Введенные данные некорректны». Такой ряд манипуляций позволит системе не пропустить некорректную информацию. Далее следует сохранение этих данных сначала в BookingController.cs, а затем в специализированном классе Booking.cs. Но этого недостаточно – необходимо отразить изменения в базе данных. На данном этапе ждет новое ветвление, если же данные успешно сохранились, то будет отправлено сообщение об успехе, а иначе – о том, что данные не сохранены. После этого всею информацию о бронировании передадут в контроллер бронирования, а оттуда на форму деталей о брони (BookingDetails.cshtml).

* 1. **Диаграмма Коммуникации**

Диаграмма коммуникации (Смотрите в приложении Б, рисунок Б.1) повторяет действия диаграммы последовательности, но имеет следующее отличие. В ней акцент сделан не на самой последовательности действий, а на взаимодействии объектов и обмене информации между ними. Здесь описан тот же процесс – бронирование номера администратором гостиницы. Но представлено это в ином виде.

* 1. **Диаграмма Развертывания**

Диаграмма Развертывания продемонстрирована на рисунке 2.2 и представляет описание аппаратной части предприятия – гостиницы. На диаграмме изображено исполнение системы. В самой верхней части располагается узел сервера базы данных гостиницы, где располагается MySQL база данных. Сервер соединен локальной сетью с сервером гостиницы. На данном сервере расположено приложение HotelServer.exe, а также расширяется артефактом файла dbadmin.exe с помощью связи deploy, где описываются методы работы с базой данной и их связями. На уровне ниже, в зависимости сервера гостиницы, организованы четыре рабочих станции: две станции администратора гостиницы с приложением HotelAdmin.exe, станция Event-менеджера с приложением HotelEventManager.exe и рабочая станция PR-менеджера с приложением HotelPRManager.exe. Все узлы рабочих станций подключены локальной сетью гостиницы – в виде связей. Так же оформлен комментарий (в виде загнутого листа), где как раз и описано подключение сервера и рабочих станций в гостинице. Помимо этого, к серверу гостиницы по закрытой сети подключены принтер и сканер, необходимые для работы персонала.

Рисунок 2.2 – Диаграмма Развертывания

* 1. **Диаграмма Деятельности**

В диаграмме деятельности (смотрите Приложение В, рисунок В.1) описан процесс бронирования номера. Процесс разделяет две линии – одна линия действий системы, другая – администратора. Действие начинается с узла начала деятельности, а сразу за ним располагается первое действие системы – отобразить список номеров. Как только клиент (администратор) увидит список будет задан вопрос – есть ли свободные номера в гостинице. Здесь свою функцию выполняет узел принятия решения, который даст выбор на два события – да или нет. Таким образом, при отсутствии номеров деятельность будет завершена узлом конца деятельности. Иначе – клиент выберет это номер, а система предложит узнать информацию о нем. Если же гостю не подходит данный номер, то можно будет выбрать номер снова (уходящая стрелка в предыдущий узел принятия решения), иначе будет предложено выполнить сразу два действия (вилка) – выполнить предварительное бронирование и заполнить данные гостя – после этого выполнится слияние. Следующим действием будет снова принятие решения – если данных клиента недостаточно, то бронь будет отменена, и следовательно, завершена деятельность. Но если все данные присутствуют, то бронирование завершается. Система предложит посмотреть и проверить все данные гостя и о номере – и если данные будут неверны, администратор внесет правки, а если верны, то бронирование завершено.

* 1. **Диаграмма Состояний**

Диаграмму состояний (смотрите в Приложение Г, рисунок Г.1) можно назвать упрощенной диаграммой деятельности, т.к. они содержат в себе одни элементы и логику. Единственным отличием является отсутствие линий системы и клиента. В этой диаграмме все действия представлены последовательно в одной системе.

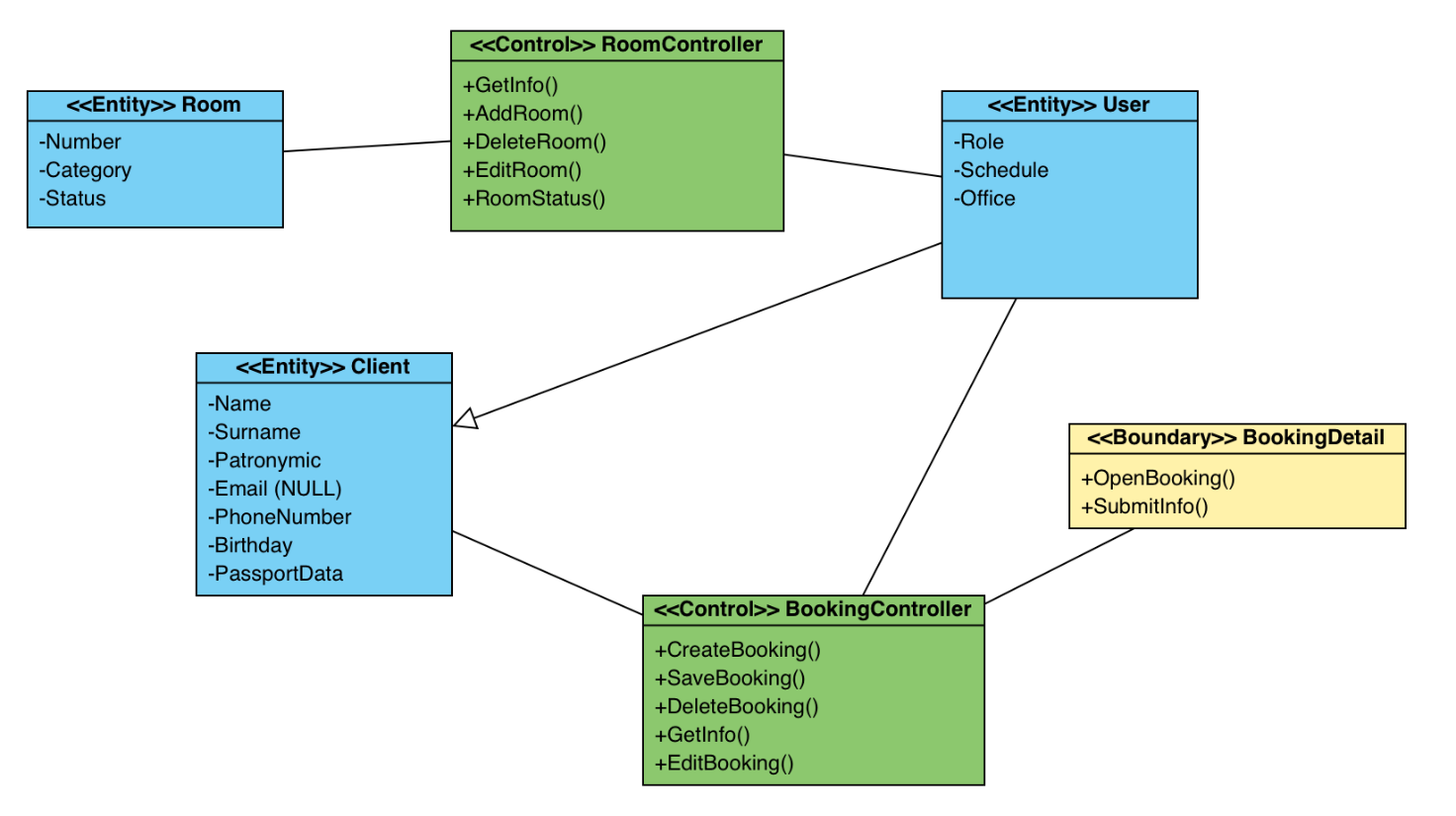
* 1. **Диаграмма Классов**

В диаграмме классов, изображенной на рисунке 2.3, необходимо определить классы нашей системы, уточняя – классы системы гостиницы. Классы бывают трех видов: управляющие – выделены зеленым цветом на диаграмме, граничные – выделены желтым и сущности – выделены голубым цветом. Таким образом становится удобнее отличать различные классы в диаграмме.

В сущностях были определены три класса: Room, User и Client. В классе Room (данные о комнате) определены поля номера комнаты, его категория и статус. В классе Client (данные о клиенте) уже определены поля имени, фамилии, отчества, электронной почты, номера телефона, даты рождения и паспортных данных гостя. В классе User (используется для работников гостиницы) определено всего три поля – роль, расписание и место работы, но т.к. используется связь обобщения, то выходит так, что все необходимые поля класса Client будут наследоваться к этому классу. В графической нотации связь обобщения изображается линией с стрелкой, пустой внутри.

После того, как описаны сущности системы – необходимо определить управляющие классы, в которых будут описаны методы для взаимодействия с данными. В классе RoomController (контроллер комнат гостиницы), связанным с классами Room и User, описаны методы получения информации о комнате, её добавления, удаления и изменения, а также получения статуса комнат для упрощения процесса поиска свободных комнат. В классе BookingController (контроллер бронирования) описаны методы для бронирования такие как: создать бронирование, сохранить бронирование, удалить, изменить и просмотреть данные о нем. Этот класс связан с классом Client, User и BookingDetail, являющимся граничным.

Рассмотрев управляющие классы и сущности, необходимо описать граничные классы, т.е. для работы непосредственно с клиентом. В классе BookingDetail (детали о бронировании) были определены всего два метода: открыть данные о бронировании и отправить их. Данный класс будет работать с представлениями, осуществляя контакт пользователя и системы.

Рисунок 2.3 – Диаграмма Классов

* 1. **Диаграмма Компонентов**

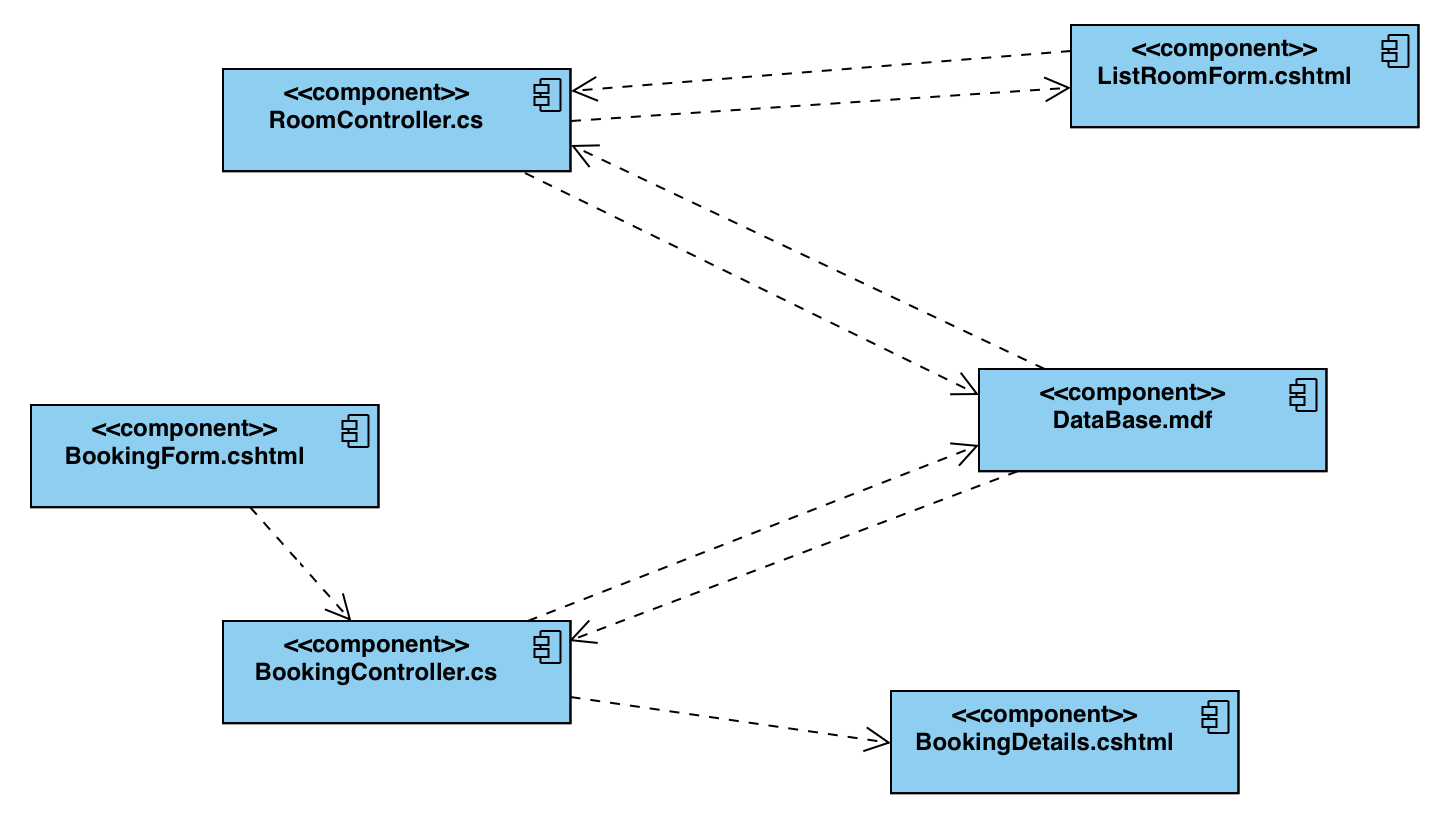
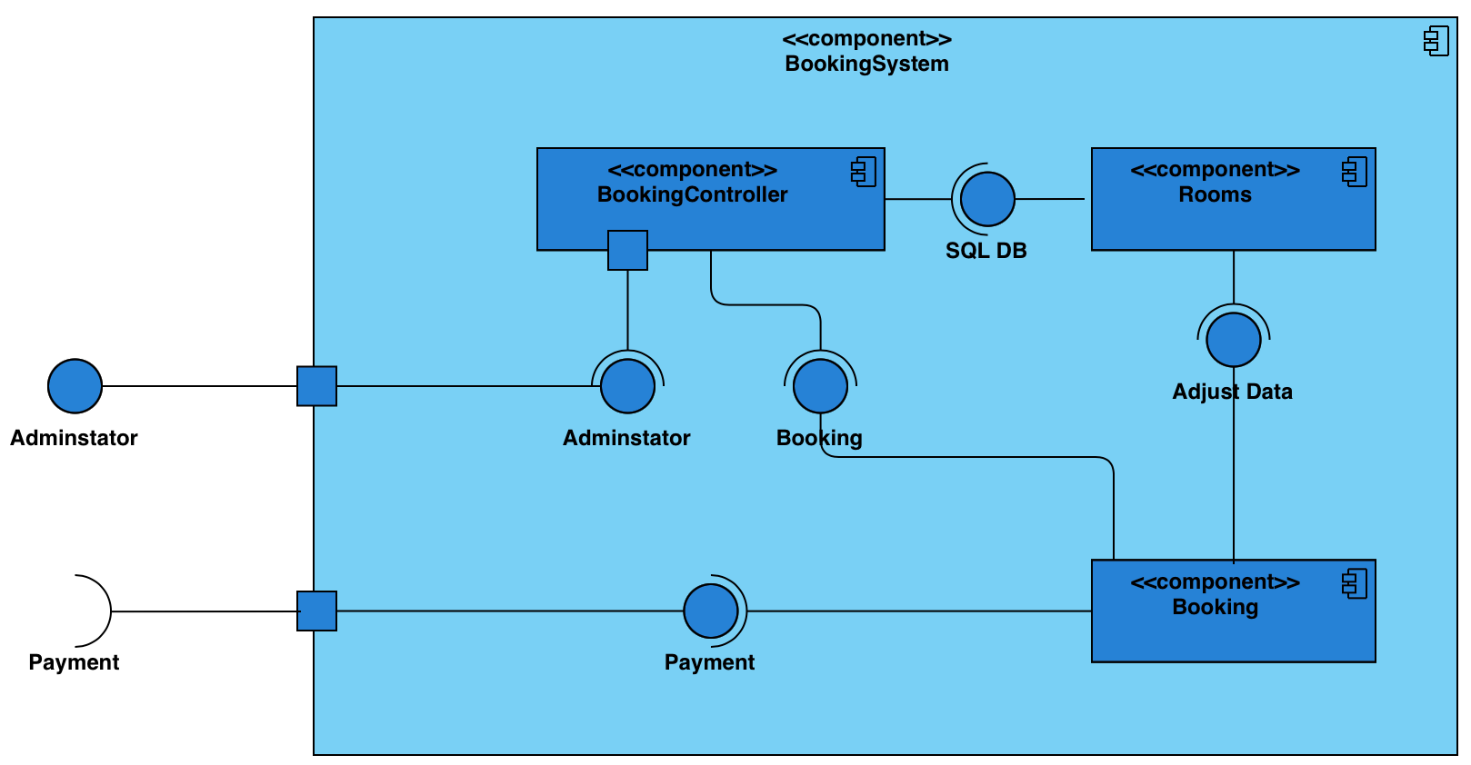
Диаграмма компонентов описывает высокоуровневое взаимодействие между компонентами системы. На рисунке 2.4 изображена система, состоящая из форм, котроллеров и базы данных. Так, между ними существуют зависимости, которые определяют взаимодействие между ними. Класс RoomController взаимодействует с формой ListRoomForm, получая и отправляя туда определенные данные. Та же зависимость и с базой данных.

Рисунок 2.4 – Диаграмма Компонентов

При описании класса BookingController можно сделать следующие выводы, класс получает данные с формы BookingForm и отправляет их в форму BookingDetail, при этом постоянно взаимодействуя с базой данных: то получая данные, то делая запросы.

На рисунке 2.5 изображена уже другая система, направленная на бронирование. В самом левом крае расположено два интерфейса: администратора и оплаты, необходимые для взаимодействия с элементами, расположенными внутри системы. Но чтобы была доступна такая возможность, необходимо допустить их вход в систему – для чего и необходимы порты. Внутри системы располагается три компонента – контроллер, класс комнат и бронирования. Таким образом, предусмотренный интерфейс на входе преобразуется в требуемый и начинает взаимодействие с контроллером, откуда на выходе образуется интерфейс Booking, требуемый для своего компонента. Но, помимо этого, компонент котроллера и класса комнат связаны с базой данной, представленной в интерфейсе, для получения необходимых данных. Следует отметить, что каждый компонент должен быть связан с своим интерфейсом. Интерфейс оплаты необходим для класса бронирования, в свою очередь, интерфейс расходных данных связан с классом комнат. Так получается полная взаимосвязь компонентов в системе.

­­­­Рисунок 2.5 – Диаграмма Компонентов

**Заключение**

В результате работы над диаграммами на тему АРМ «Гостиница» были закреплены знания по дисциплине «Технология разработки программного обеспечения» и работе с онлайн-сервисами и приложением Microsoft Visio. Следует отметить, что были приобретены новые знания и умения по работе с различными видами диаграмм, по работе с источниками и информацией.

Все поставленные задачи курсового проекта были выполнения. По итогу выполнения работы были спроектированы следующие диаграммы: Вариантов Использования, Последовательности, Коммуникации, Развертывания, Деятельности, Состояний, Классов и Компонентов. Следует отметить, что была затронута очень важная и актуальная тема на данный момент – туризм с каждым годом развивается стремительнее, но многие гостиницы оказываются неготовыми к большому потоку клиентов. Поэтому необходимо создать удобные и простые системы для работы персоналу, фундамент которого заложен в данных диаграммах.

UML является идеальным средством для проектирования системы, которую потом можно масштабировать. Благодаря нему возможно посмотреть на все аспекты с разных точек зрения, что является самым важным преимуществом. Он дает возможность увидеть недостатки системы и снизить нагрузку на определенные компоненты. Так, появляется возможность рассмотреть самые большие системы от вариантов использования и описания классов до аппаратной составляющей.

Овладев опытом проектирования диаграмм, следует сделать вывод, что все системы начинаются с их проектирования. Данная работа является невероятным опытом в разработке приложений, систем, баз данных и т.д., которой станет началом в большом будущем в профессии программиста.

**Список использованной литературы**

1. WorldSkills Russia [Электронный ресурс]. – Режим доступа: https://nationalteam.worldskills.ru/skills/programmnye-resheniya-dlya-biznesa/;
2. Сайт о программировании C+ [Электронный ресурс]. – Режим доступа: https://prog-cpp.ru/uml-classes/;
3. FlexBerry [Электронный ресурс]. – Режим доступа: https://flexberry.github.io/ru/fd\_activity-diagram.html;
4. Сообщество программистов. Хабр [Электронный ресурс]. – Режим доступа: https://habr.com/ru/post/566218/;
5. Учебная и научная деятельность Анисимова Владимира Викторовича [Электронный ресурс]. – Режим доступа: https://sites.google.com/site/anisimovkhv/learning/pris/lecture/tema11;
6. IT-GOST [Электронный ресурс]. – Режим доступа: http://it-gost.ru/articles/view\_articles/94;
7. Роберт А. Максимирчук, Эрик Дж. Нейбург. UML для простых смертных. [Текст] – Лори, 2021, - 270 стр.

**Приложение А**

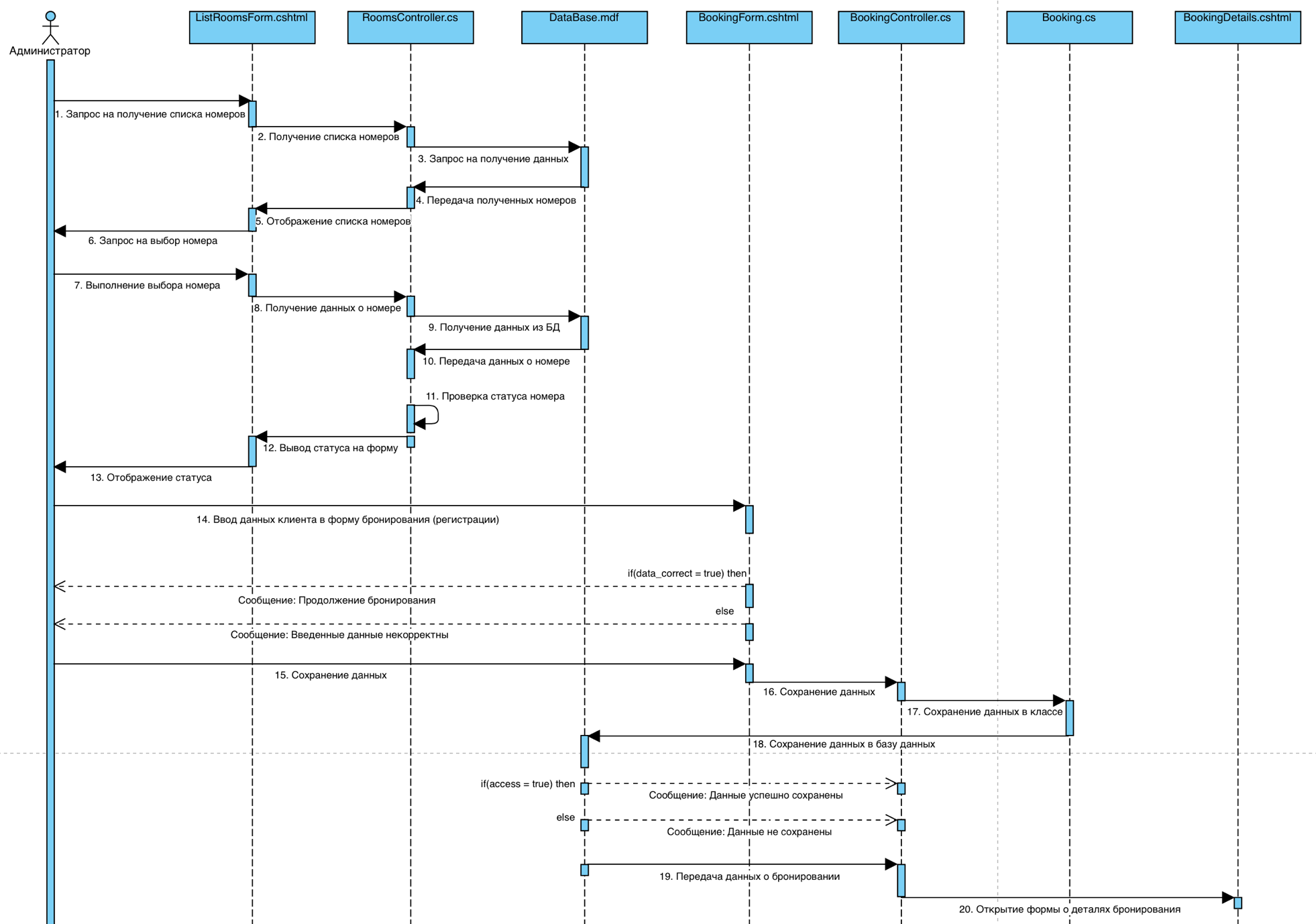
Диаграмма Последовательности

Рисунок А.1 – Диаграмма Последовательности

**Приложение Б**

Диаграмма Коммуникации

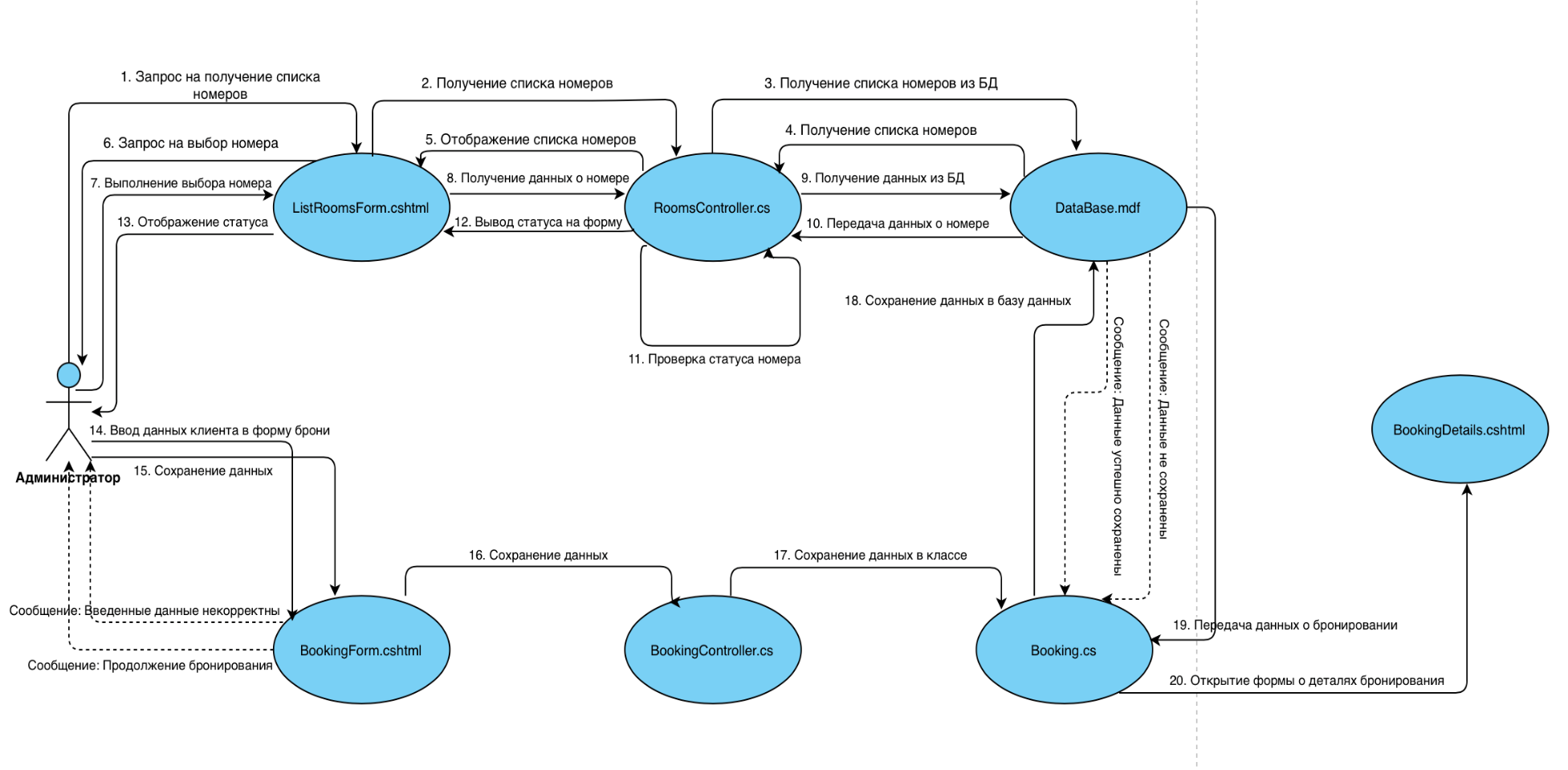


Рисунок Б.1 – Диаграмма Коммуникации

**Приложение В**

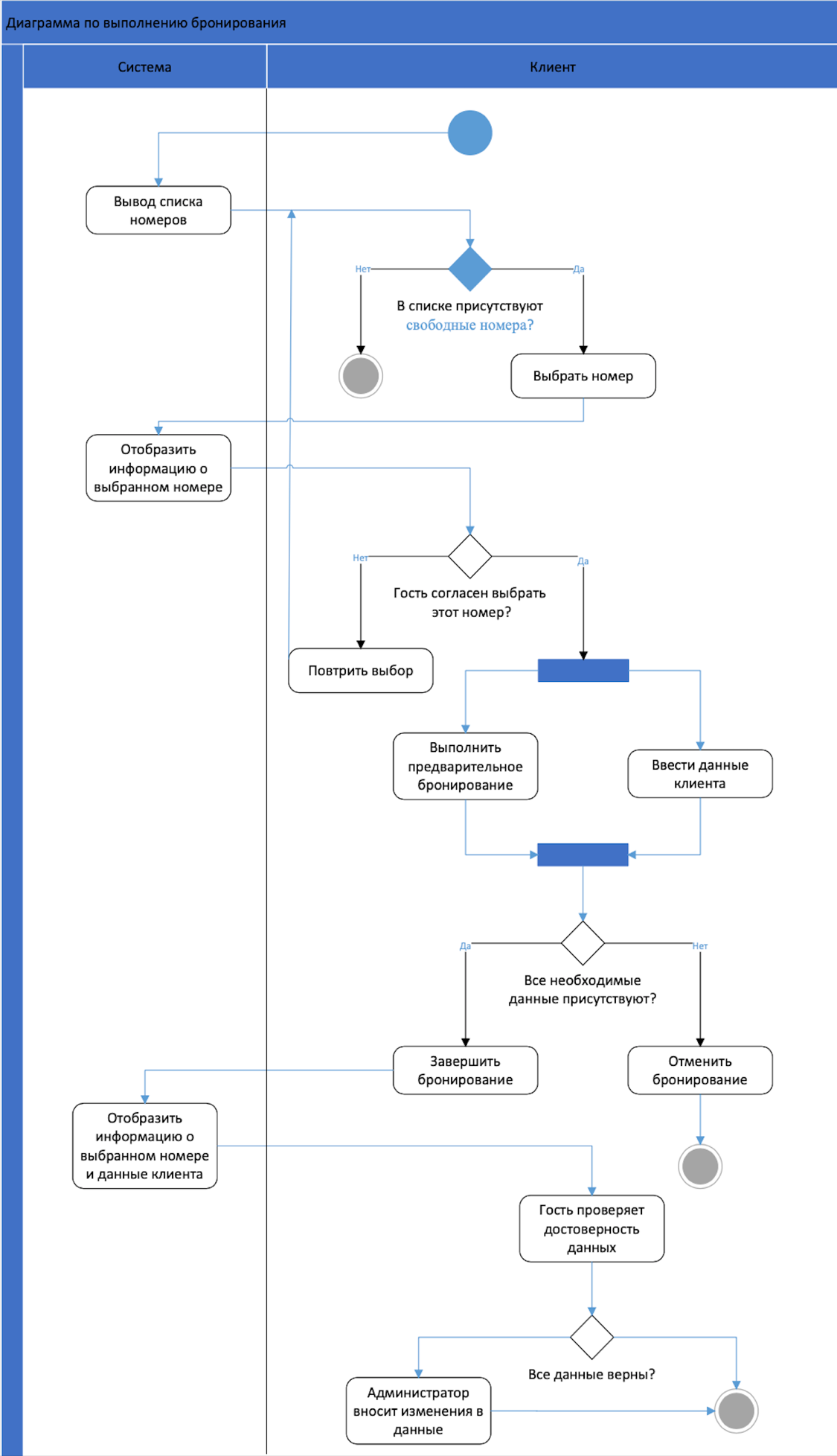
Диаграмма Деятельности

Рисунок В.1 – Диаграмма Последовательности

**Приложение Г**

Диаграмма Состояний

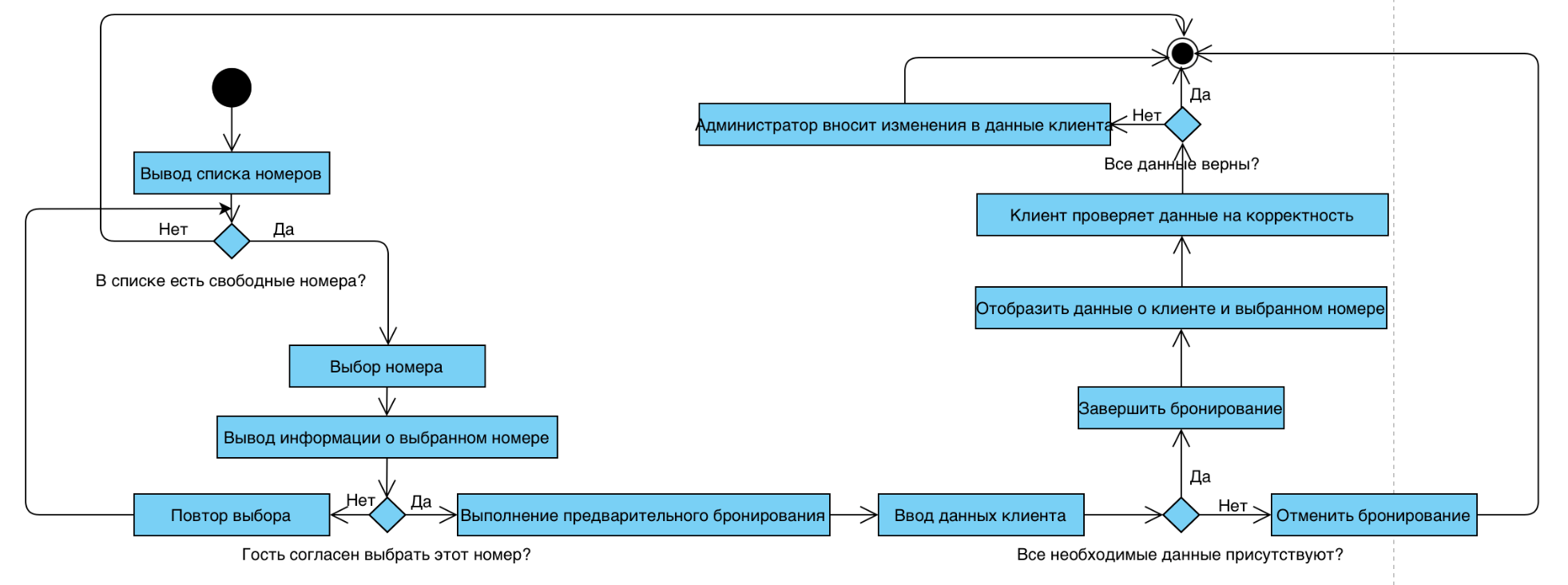


Рисунок Г.1 – Диаграмма Состояний