Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

**Отчёт**

по предмету «Проектирование программного обеспечения»

Лабораторная работа №7

«Интерфейс поиска и бронирования лекарств»

Студент: Немкович А. В.

ФИТ 3 курс 1 группа

Преподаватель: Якубенко К. Д

Минск 2024

# Введение

Проектируемая модель представляет собой веб-приложение, которое предоставляет сервис для поиска, бронирования и покупки лекарственных средств в аптеках города. Основная цель проекта — упростить процесс поиска медикаментов и повысить их доступность для пользователей через удобный и интуитивно понятный интерфейс.

Сервис ориентирован на несколько категорий пользователей: гостей, зарегистрированных клиентов, представителей аптек и менеджеров. Каждый пользовательский сегмент имеет доступ к набору функций, которые помогают решать их задачи.

Гости, которые не авторизованы в системе, могут выполнять поиск лекарств по названию, активному веществу или другим параметрам. Им доступна информация о наличии медикаментов в аптеках, ценах, а также подробные описания, включая инструкции по применению и возможные побочные эффекты. Также пользователи могут получить сведения о ближайших аптеках, их местоположении, времени работы и контактных данных.

Зарегистрированные пользователи дополнительно могут бронировать лекарства для самовывоза, сохранять медикаменты в списке избранного и управлять своими заказами в личном кабинете. Эти функции повышают удобство использования платформы и предоставляют персонализированный опыт.

Представители аптек имеют возможность обновлять данные о товарах, их ценах и наличии. Также им предоставляется инструмент для управления заказами и бронированиями через административную панель. Это помогает аптечным сетям поддерживать актуальность информации и улучшать качество обслуживания.

Менеджеры системы выполняют задачи по управлению ресурсами сервиса. Они могут добавлять и редактировать информацию о лекарствах, собирать статистику использования приложения и предоставлять отчеты.

Сервис использует многоуровневую архитектуру, обеспечивая разделение обязанностей между клиентской и серверной частями. Клиентская часть представлена веб-интерфейсом, доступным через любой современный браузер. Серверная часть построена на ASP.NET Core, который отвечает за обработку запросов и взаимодействие с базой данных Oracle через Entity Framework.

# 1.Описание программных средств

## 1.1. Описание программного средства Figma

Название: Figma.

Версия: Последняя доступная версия обновляется разработчиком автоматически, так как Figma является облачным приложением, постоянно получающим улучшения и обновления. Версии не фиксируются как у традиционного программного обеспечения, но можно узнать текущее состояние приложения в документации и на официальном сайте.

Разработчик: Figma, Inc.

Адрес загрузки: https://www.figma.com

Режим использования: Figma — это кросс-платформенное облачное приложение, основная работа в котором ведется в режиме онлайн через веб-браузер. Это одно из первых профессиональных средств для UI/UX-дизайна, полностью основанное на облачных технологиях, что позволяет пользователям работать с проектами без установки тяжелых приложений на компьютер.

Доступность на платформах: Веб-приложение доступно на всех устройствах с поддержкой современных браузеров, таких как Google Chrome, Mozilla Firefox, Safari, Microsoft Edge. Настольные версии поддерживаются на Windows и macOS, что позволяет пользователям работать с проектами как в режиме онлайн, так и частично офлайн. Приложения для iOS и Android доступны для просмотра проектов и прототипов.

Figma работает с различными моделями интерфейсов и дизайнов:

* 2D-векторные модели. Позволяет создавать элементы интерфейсов, кнопки, иконки, макеты страниц и другие элементы интерфейсов, основанные на векторной графике.
* Прототипы. Поддерживает создание интерактивных макетов пользовательских интерфейсов, где можно моделировать сценарии использования приложений и сайтов, а также тестировать навигацию и взаимодействие.

Одна из главных сильных сторон Figma — это возможность многопользовательского режима редактирования в реальном времени. Несколько человек могут одновременно вносить изменения в проект, оставлять комментарии и общаться внутри интерфейса, что ускоряет процесс работы и упрощает координацию команд.

# 2. Описание практического задания

Первой диаграммой была выбрана диаграмма деятельности – рисунок 2.1. Она посвящена процессу поиска лекарства и где его можно приобрести.

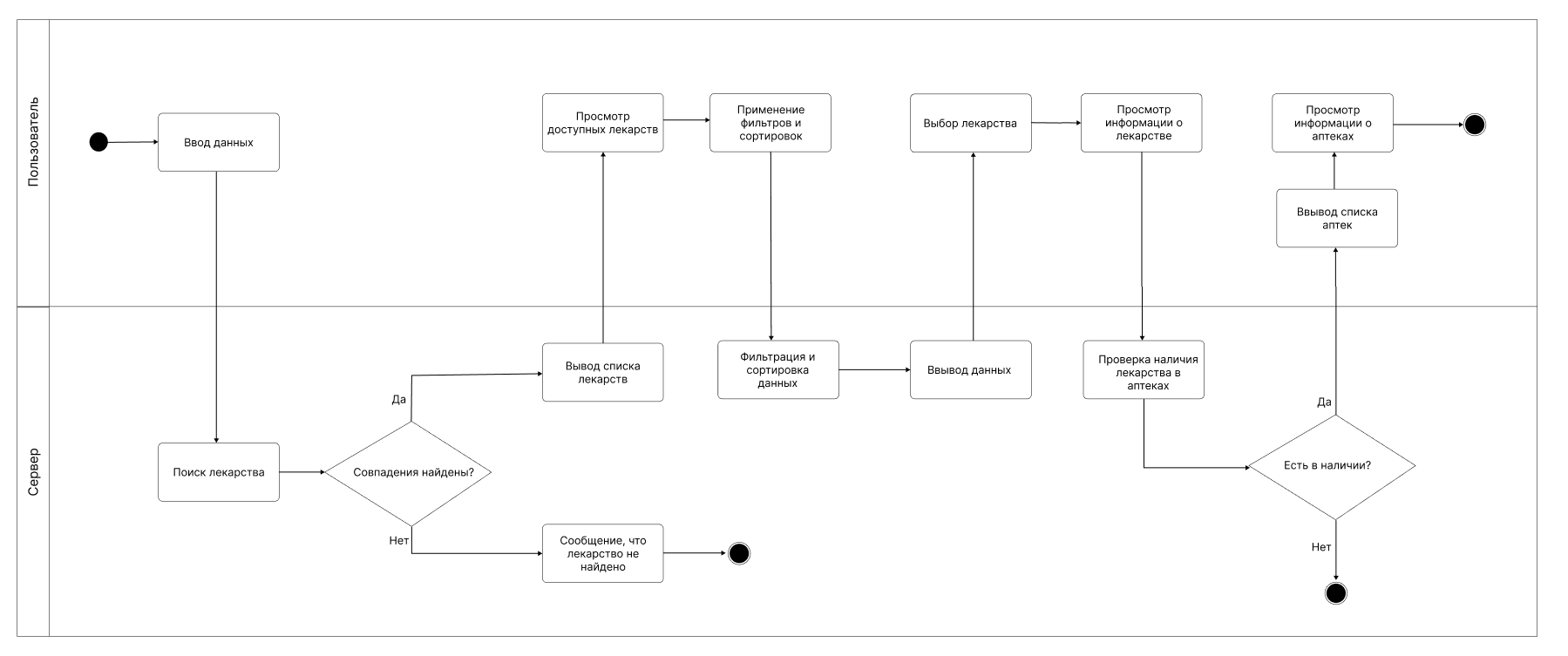


Рисунок 2.1 – Диаграмма деятельности

Диаграмма деятельности должна включать основные элементы, такие как действия, условия, начальные и конечные точки, а также зоны ответственности (плавательные дорожки). Действия отображаются в виде прямоугольников и обозначают операции, выполняемые либо пользователем, либо системой. Например, такие операции, как ввод данных, поиск лекарства, применение фильтров и сортировок, выбор лекарства, проверка наличия лекарства в аптеке, должны быть обозначены как последовательные шаги процесса.

Для представления условий используются ромбы. Они позволяют ветвить процесс в зависимости от результата проверки. Например, условие "Лекарство найдено?" имеет два возможных выхода: "Да" и "Нет". При положительном результате процесс продолжается выводом списка лекарств, при отрицательном — выводом сообщения о том, что лекарство не найдено.

Начальная точка процесса обозначается черным залитым кругом и размещается в начале диаграммы, указывая на начало выполнения. Конечная точка процесса отображается черным кругом с белым контуром, который завершается в конце каждой ветви диаграммы.

Для разделения зон ответственности используются плавательные дорожки. Они показывают, какие действия выполняются пользователем, а какие — системой. Например, ввод данных и выбор фильтров — это действия пользователя, а поиск лекарства в базе данных и проверка наличия — задачи системы. Потоки управления между действиями обозначаются стрелками, которые указывают последовательность выполнения и переходы между этапами.

Вторая диаграмма – диаграмма состояния. Отобразим на ней процесс бронирования лекарства.

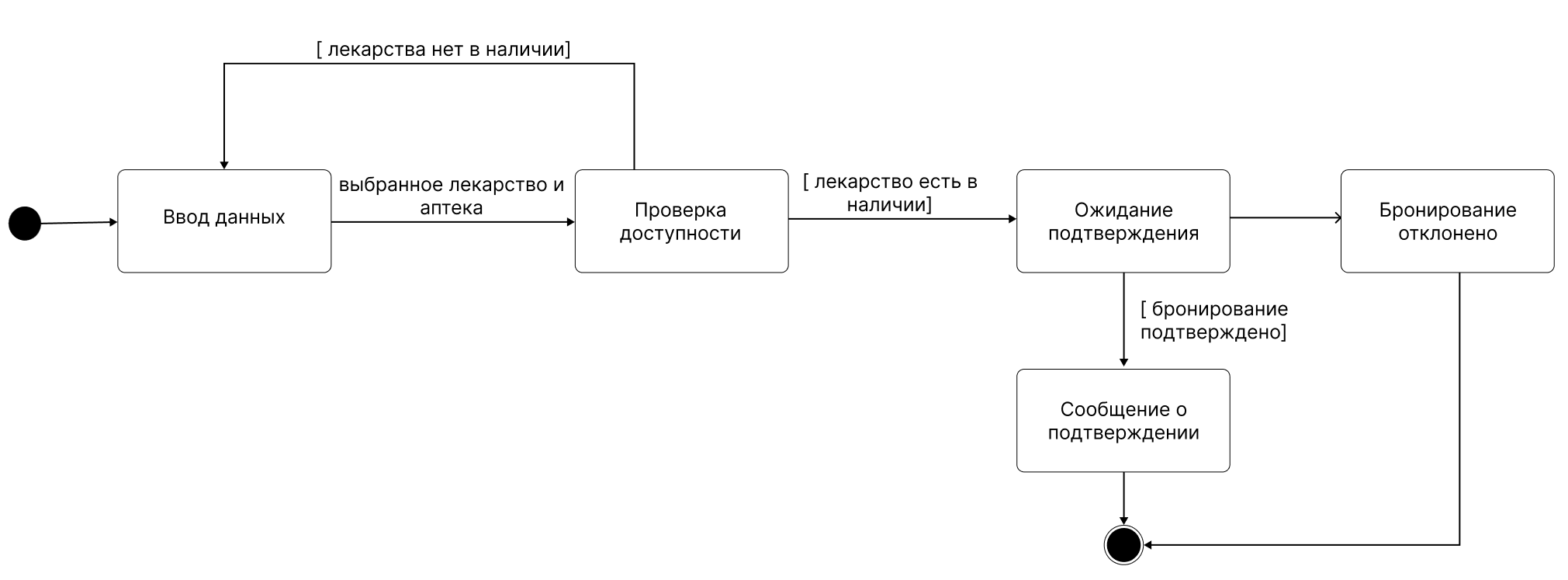


Рисунок 2.2 – Диаграмма состояния

Диаграмма состояния для процесса бронирования включает основные элементы, которые отражают этапы и возможные переходы между ними. Начальным элементом является точка, обозначающая начало процесса, от которой начинается переход к состоянию "Ввод данных". На этом этапе пользователь выбирает необходимое лекарство и аптеку. После ввода данных система переходит в состояние "Проверка доступности", где анализируется наличие лекарства в указанной аптеке.

Далее происходит разветвление на два возможных исхода. Если лекарство доступно, процесс продолжается в состоянии "Ожидание подтверждения", где запрос направляется представителю аптеки для подтверждения бронирования. Если же лекарство отсутствует, система переходит к состоянию "Бронирование отклонено", что завершает процесс с уведомлением пользователя.

После состояния "Ожидание подтверждения" возможно два пути. Если представитель аптеки подтверждает бронирование, система переходит в состояние "Сообщение о подтверждении", где пользователь уведомляется об успешной обработке бронирования. В случае отклонения бронирования процесс завершается состоянием "Бронирование отклонено".

Все состояния и переходы соединены направленными стрелками, которые показывают последовательность действий и взаимосвязи между этапами. Условные переходы, такие как [лекарство есть в наличии] или [лекарства нет в наличии], задают направление между состояниями в зависимости от результатов проверки или действий представителя аптеки. Завершается диаграмма конечным состоянием, обозначающим завершение всего процесса.

# 3.Ответы на вопросы

## 3.1. Укажите виды диаграмм поведения. Какая между ними связь?

Диаграмма деятельности (Activity Diagram) описывает поток управления или выполнения действий внутри системы. Она используется для моделирования бизнес-процессов, алгоритмов или любых других процессов, где важно понять последовательность действий и переходы между ними. Диаграмма активности наглядно отображает, как различные действия могут быть параллельными или зависимыми друг от друга. Она также может включать условия, циклы и другие элементы для моделирования сложных процессов.

Диаграмма последовательности (Sequence Diagram) фокусируется на взаимодействии объектов через время. Это один из видов диаграмм взаимодействия, который показывает последовательность сообщений, передаваемых между объектами. Она позволяет понять, как данные передаются между объектами в ответ на запросы и события. Диаграмма последовательности особенно полезна для анализа сценариев использования, так как она отображает, как система будет реагировать на конкретные запросы.

Диаграмма состояний (State Diagram) моделирует различные состояния объекта и переходы между ними, которые происходят в ответ на события. Она полезна для описания поведения объектов, которые могут находиться в разных состояниях в течение своего жизненного цикла. Это может быть полезно, например, для моделирования объектов с длительным временем жизни или сложным состоянием, таких как процессы бронирования, покупок или пользовательские сессии.

## 3.2. Опишите назначение диаграммы деятельности.

Диаграмма деятельности используется для моделирования рабочих процессов и бизнес-процессов в системе, а также для отображения последовательности действий, выполняемых в ходе какого-либо процесса. Она помогает наглядно представить, как выполняются задачи, какие решения принимаются в процессе, как данные передаются между действиями, и как они могут быть выполнены параллельно или последовательно. Диаграмма деятельности отображает как основной поток действий, так и альтернативные пути в зависимости от условий.

## 3.3. Опишите основные нотации, которые используются на диаграмме состояний.

Состояние (State): Прямоугольник с закругленными углами, представляющий состояние объекта в какой-то момент времени. Состояния могут быть простыми или составными (состояния, состоящие из других состояний).

Начальное состояние (Initial State): Круглый символ с заполненной точкой внутри, обозначает начало жизненного цикла объекта или процесса.

Конечное состояние (Final State): Круг с маленьким кружком внутри, который указывает на завершение процесса или завершение жизненного цикла объекта.

Переход (Transition): Линия с стрелкой, которая соединяет два состояния, показывая возможный переход из одного состояния в другое, обычно в ответ на событие или условие.

Событие (Event): Меченое на стрелке перехода событие, которое вызывает изменение состояния.

Условие (Guard Condition): Условия, которые должны быть выполнены для того, чтобы переход состоялся. Обычно они записываются в квадратных скобках рядом с переходом.

История (History): Специальный символ, который используется для обозначения сохранения последнего состояния, когда объект возвращается в предыдущий контекст.

## 3.4. Укажите виды связей между объектами на диаграмме последовательностей.

Сообщение (Message): Основная связь между объектами, которая передает данные или вызывает операцию. Сообщения могут быть синхронными (когда объект ожидает ответа) или асинхронными (когда объект не ожидает ответа и продолжает выполнение).

Вызов операции (Call): Сообщение, которое вызывает выполнение операции или метода на другом объекте. Обычно обозначается стрелкой с маркировкой, указывающей на операцию.

Ответ (Return): Ответ на вызов операции, который обычно возвращает какое-либо значение или подтверждает выполнение операции.

Создание объекта (Object Creation): Связь, показывающая создание нового объекта, который может быть использован в ходе взаимодействия.

Уничтожение объекта (Object Destruction): Связь, указывающая на уничтожение объекта в процессе взаимодействия.

## 3.5. Какая диаграмма позволяет моделировать параллельные вычисления?

Для моделирования параллельных вычислений в UML используется диаграмма деятельности. Этот вид диаграммы позволяет отображать параллельные потоки выполнения, которые могут происходить одновременно. В диаграмме деятельности используется нотация для параллельных потоков, такие как параллельные разветвления (fork) и параллельные слияния (join), которые разделяют и объединяют потоки выполнения, что помогает моделировать процессы, которые могут выполняться одновременно.