Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ   
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина: Модели данных и системы управления базами данных

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

на тему

**ПРОГРАММНОЕ СРЕДСТВО ДЛЯ ВЕБ-СЕРВИСА ДОСТАВКИ ЕДЫ**

БГУИР КП 1-40 04 01 014 ПЗ

|  |  |  |
| --- | --- | --- |
| Студент |  | А.С. Жагоров |
| Руководитель |  | А.В. Давыдчик |
|  |  |  |

Минск 2024

**СОДЕРЖАНИЕ**

Введение 3

1 Архитектура вычислительной системы 4

1.1 Структура и архитектура вычислительной системы 4

1.2 История, версии и достоинства 6

1.3 Обоснование выбора вычислительной системы 7

2 Платформа программмного обеспечения 9

2.1 Выбор операционной системы 9

2.2 Выбор платформы для написания программы 9

3 Теоретическое обоснование разработки программного продукта 11

3.1 Обоснование необходимости разработки. 11

3.2 Технологии программирования, используемые для решения поставленных задач. 12

4 Проектирование функциональных возможностей программы 20

4.1 Настройка взаимодействия сервера и базы данных 20

4.2 Настройка взаимодействия клиента и сервера 21

5 Проектирование разрабатываемой базы данных программного  
 обеспечения 22

Заключение 29

Список литературных источников 30

Приложение А (обязательное) Листинг программного кода 32

Приложение Б (обязательное) Конечная схема базы данных 38  
Приложение В (обязательное) Ведомость курсового проекта 39

# **ВВЕДЕНИЕ**

Аренда стала неотъемлемой частью современного общества, позволяя людям временно пользоваться необходимыми вещами, недвижимостью или услугами без необходимости их приобретения. На данный момент рынок аренды представлен отдельными приложениями, которые специализируются на узких отраслях, таких как аренда недвижимости, автомобилей, студийных пространств или одежды. Отсутствие единого пространства для аренды ограничивает возможности пользователей.

Создание универсального арендного маркетплейса способно решить эту проблему, объединив в одной платформе арендаторов и арендодателей. Это позволит повысить удобство поиска и аренды, а также обеспечит дополнительные возможности для продвижения и рекламы объектов. Особенно актуальна такая система для малого и среднего бизнеса, который сталкивается с трудностями в поиске подходящих инструментов для управления бронированиями.

Данный курсовой проект направлен на разработку программного обеспечения для арендного маркетплейса, где физические и юридические лица смогут арендовать практически всё: от недвижимости и автомобилей до студийных пространств и одежды в шоурумах. Основная цель разработки – создание единой платформы, которая обеспечит удобный процесс поиска, бронирования и взаимодействия между арендаторами и арендодателями.

В соответствии с поставленной целью проекта определены следующие ключевые задачи:

1 Определить сущности, которые относятся к предметной области аренды.

2 Разработать структуру базы данных, охватывающую все аспекты функционирования платформы, включая управление объявлениями, бронированиями и пользовательской информацией.

3 Создать приложение, которое будет использовать разработанную базу данных для предоставления пользователям удобного интерфейса.

Реализация данной системы упростит рабочие процессы в сфере аренды, оптимизирует взаимодействие между участниками платформы и повысит уровень сервиса. Конечный продукт должен стать удобным инструментом как для арендаторов, так и для арендодателей, способствуя развитию арендного рынка и удовлетворению потребностей всех его участников.

# АРХИТЕКТУРА ВЫЧИСЛИТЕЛЬНОЙ СИСТЕМЫ

## 1.1 Структура и архитектура вычислительной системы

PostgreSQL – это объектно-реляционная система управления базами данных (ORDBMS), наиболее развитая из открытых СУБД в мире. Имеет открытый исходный код и является альтернативой коммерческим базам данных.

Ранние версии системы были основаны на старой программе POSTGRES University, созданной университетом Беркли: так появилось название PostgreSQL. Существуют сокращения PSQL и PgSQL — они тоже обозначают PostgreSQL.

СУБД позволяет гибко управлять базами данных (БД). С ее помощью можно создавать, модифицировать или удалять записи, отправлять транзакцию — набор из нескольких последовательных запросов на особом языке запросов SQL.

Для чего нужна PostgreSQL:

1) Гибкий доступ к базам данных, их организация и хранение;

2) Управление записями в базах данных;

3) Просмотр нужной информации из базы по запросу;

4) Отправка транзакций, последовательных запросов;

5) Настройка и контроль доступа к той или иной информации;

6) Защита информации от возможных утечек и потерь.

Иногда СУБД называют бесплатным аналогом Oracle Database. Обе системы адаптированы под большие проекты и высокую нагрузку. Но есть разница: они по-разному хранят данные, предоставляют разные инструменты и различаются возможностями.

Традиционно популярные СУБД – реляционные. Это значит, что данные, которые в них хранятся, представляются в виде записей, связанных друг с другом отношениями, – relations. Получаются связанные списки, которые могут иметь между собой те или иные отношения, – так и образуется таблица.

Существует еще одна популярная модель – объектная. Данные представляются в виде объектов, их атрибутов, методов и классов. Объектная модель поддерживает возможности, о которых мы подробно рассказывали в статье про ООП, например наследование.

PostgreSQL – объектно-реляционная СУБД. Это значит, что она поддерживает и объектный, и реляционный подход.

Еще одна особенность PostgreSQL – поддержка большого количества типов записи информации. Это не только стандартные целочисленные значения, числа с плавающей точкой, строки и булевы значения, но и денежный, геометрический, перечисляемый, бинарный и другие типы. PostgreSQL по умолчанию поддерживает битовые строки и сетевые адреса, массивы данных, в том числе многомерные, композитные типы и другие сложные структуры. В ней есть поддержка XML, JSON и NoSQL-баз.

При необходимости к СУБД можно подключить поддержку типов данных, которые нужны в конкретном проекте. В PostgreSQL есть несколько внутренних форматов, которые используются только в ней.

В большинстве СУБД, рассчитанных на средние и небольшие проекты, есть ограничения по объему базы и количеству записей в ней. В PostgreSQL ограничений нет.

Ограничения касаются только конкретных записей. Одна таблица может занимать не больше 32 Тб, а одна запись – 1,6 Тб. В одном поле записи может быть не больше 1 Гб данных, а максимальное количество полей зависит от типа и составляет от 250 до 1600 штук. Максимальных значений хватает, чтобы хранить в БД любые данные.

PostgreSQL работает со сложными, составными запросами. Система справляется с задачами разбора и выполнения трудоемких операций, которые подразумевают и чтение, и запись, и валидацию одновременно. Она медленнее аналогов, если речь заходит только о чтении, но в других аспектах превосходит конкурентов.

В PostgreSQL можно писать собственные функции – пользовательские блоки кода, которые выполняют те или иные действия. Эта возможность есть практически в любых СУБД, но PostgreSQL поддерживает больше языков, чем аналоги. Кроме стандартного SQL, в PostgreSQL можно писать на C и C++, Java, Python, PHP, Lua и Ruby. Он поддерживает V8 – один из движков JavaScript, поэтому JS тоже можно использовать совместно с PgSQL. Реализована поддержка Delphi, Lisp и прочих редких языков. При необходимости можно расширить систему под другие ЯП.

Модификация SQL, которая используется в PostgreSQL, называется PL/pgSQL. Это процедурное расширение, которое поддерживает сложные вычисления и дополняет «классический» SQL новыми возможностями. Одновременная модификация базы. Важная особенность PostgreSQL – возможность одновременного доступа к базе с нескольких устройств. В СУБД реализована клиент-серверная архитектура, когда база данных хранится на сервере, а доступ к ней осуществляется с клиентских компьютеров. Так, например, реализуются разнообразные сайты. Одна из возможных сложностей – ситуация, когда несколько человек одновременно модифицируют базу и нужно избежать конфликтов.

В PostgreSQL для этого используется технология MVCC – Multiversion Concurrency Control, многоверсионное управление параллельным доступом. Каждый пользователь получает снапшот базы, в который вносятся изменения. Только после фиксации транзакции они поступают в исходную базу данных. Пока человек вносит изменения, они не видны другим пользователям. Конфликтов не возникает, как и необходимости блокировать чтение или запись.

PostgreSQL соблюдает требования ACID благодаря технологии MVCC. Это делает систему надежной и безопасной в использовании, а данные – защищенными от возможных сбоев, ошибок и потерь.

## История, версии и достоинства

PostgreSQL создана на основе некоммерческой СУБД Postgres, разработанной как open-source проект в Калифорнийском университете в Беркли. К разработке Postgres, начавшейся в 1986 году, имел непосредственное отношение Майкл Стоунбрейкер, руководитель более раннего проекта Ingres, на тот момент уже приобретённого компанией Computer Associates.

Стоунбрейкер и его студенты разрабатывали новую СУБД в течение восьми лет с 1986 по 1994 годы. За этот период в синтаксис были введены процедуры, правила, пользовательские типы и другие компоненты. В 1995 году разработка снова разделилась: Стоунбрейкер использовал полученный опыт в создании коммерческой СУБД Illustra, продвигаемой его собственной одноимённой компанией (приобретённой впоследствии компанией Informix), а его студенты разработали новую версию Postgres – Postgres95, в которой язык запросов POSTQUEL был заменен на SQL.

Разработка Postgres95 была выведена за пределы университета и передана команде энтузиастов. Новая СУБД получила имя, под которым она известна и развивается в текущий момент – PostgreSQL.

PostgreSQL обладает следующими преимуществами:

1 Возможность расширения. Разработчик может написать для СУБД собственные типы и их преобразования, операции и функции, ограничения и индексы, собственный процедурный язык для запросов. PostgreSQL можно модифицировать практически под любую нестандартную задачу.

2 Высокая мощность и широкая функциональность. PostgreSQL – возможно, единственная бесплатная СУБД с открытым исходным кодом, которая рассчитана на работу с объемными и сложными проектами. Она мощная, производительная, способна эффективно работать с большими массивами данных. Есть примеры реального использования СУБД для баз данных в несколько петабайт с сотнями тысяч запросов в секунду.

3 Открытость. PostgreSQL – ПО с открытым исходным кодом, которое распространяется по свободной лицензии. Это означает, что любой разработчик может посмотреть, как написана система, или предложить для нее свои правки. СУБД разрабатывается сообществом энтузиастов и в определенной степени никому не принадлежит, а значит, ее можно свободно и без ограничений использовать в своих проектах.

4 Минимальное количество багов. PostgreSQL – проект, который известен высоким качеством отладки. Каждая версия системы появляется в доступе только после полной проверки, поэтому СУБД очень стабильна. Частая проблема бесплатных проектов – новые версии с большим количеством багов, но в случае с PostgreSQL такой проблемы нет.

5 Согласно независимым автоматизированным исследованиям, в исходном коде СУБД есть одна ошибка на 39 000 строк кода. Это в пять раз меньше, чем в MySQL, и в пятьдесят раз меньше, чем в ядре операционной системы Linux.

6 Кроссплатформенность. Чаще всего PostgreSQL используют на серверах с операционными системами семейства Linux, но СУБД поддерживает и другие ОС. Ее можно установить в системы на базе Windows, BSD, macOS и Solaris. Кроме того, у PostgreSQL есть автономный веб-сервер PostgREST, с которым можно работать с помощью REST API. СУБД можно развернуть и в облаке.

PostgreSQL не имеет однозначных недостатков. Она подходит для проектов, где нужна мощная база данных с высокой функциональностью. Но для большинства приложений и сайтов требуется менее мощная и более быстрая, легковесная, простая в освоении СУБД. Обычно в таких случаях используется MySQL. [1]

## 1.3 Обоснование выбора вычислительной системы

Для реализации арендного маркетплейса в данном проекте была выбрана СУБД PostgreSQL благодаря её высокой надёжности, открытости и широкому набору возможностей. Эта система позволяет эффективно хранить и обрабатывать данные о пользователях, арендуемых объектах и бронированиях, что особенно важно для платформы, обеспечивающей взаимодействие между арендаторами и арендодателями. PostgreSQL обеспечивает надежность работы с информацией, что критично для сервиса, где требуется актуальность данных и безопасность личной информации.

Поддержка ACID-транзакций гарантирует целостность данных в условиях многопользовательского доступа, а развитая система разграничения прав позволяет гибко настраивать уровни доступа для разных категорий пользователей. Это особенно полезно для разделения прав арендаторов, арендодателей и администраторов платформы.

Кроме того, PostgreSQL обладает высокой масштабируемостью и совместимостью с различными операционными системами, что делает её подходящей как для локального развертывания, так и для облачных решений. Возможность расширять функционал через плагины и поддержку множества языков программирования позволяет адаптировать PostgreSQL к изменяющимся требованиям проекта и увеличению объёмов данных, обеспечивая стабильность и производительность системы в долгосрочной перспективе. [2]

# ПЛАТФОРМА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

### **2.1 Выбор операционной системы**

Для разработки системы онлайн-заказа лекарств выбрана операционная система Windows. Такой выбор обусловлен её уникальными характеристиками, обеспечивающими удобство разработки, развертывания и эксплуатации кросс-платформенных приложений.

Windows предоставляет стабильную и универсальную среду, которая идеально подходит для создания приложений с графическим интерфейсом, веб-сервисов и облачных решений. Благодаря регулярным обновлениям системы, включающим как улучшения функциональности, так и патчи безопасности, платформа остается актуальной и защищенной.

Одной из ключевых причин выбора Windows является её интеграция с экосистемой Microsoft. Это позволяет использовать единую платформу для разработки (Visual Studio), управления базами данных (SQL Server), облачного хостинга (Azure) и интеграции с другими сервисами Microsoft, такими как Active Directory и Office 365.

Кроме того, Windows предоставляет обширные возможности для разработки пользовательских интерфейсов и настольных приложений. Поддержка графических API, таких как DirectX и WinUI, делает эту платформу особенно подходящей для задач, связанных с созданием визуально насыщенных приложений.

Windows предлагает встроенные средства для управления безопасностью, включая расширенную систему прав доступа, шифрование данных и централизованный мониторинг. Это помогает обеспечить высокий уровень защиты пользовательских данных и соответствие стандартам конфиденциальности. [3]

Таким образом, выбор Windows в качестве операционной системы позволяет создать надежное и производительное решение, полностью отвечающее требованиям приложения для удобного управления коворкингами.

### **Выбор платформы для разработки**

В качестве платформы для разработки системы был выбран стек технологий Microsoft, включающий ASP.NET Core для серверной части и .NET MAUI для пользовательского интерфейса. Такой выбор обусловлен высокой производительностью, гибкостью и возможностью создания кросс-платформенных решений.

ASP.NET Core обеспечивает мощные возможности для построения веб-сервисов и API. Это высокопроизводительный и модульный фреймворк, который оптимизирован для работы в облаке и локально. Использование ASP.NET Core позволяет разработчикам легко управлять маршрутизацией, безопасностью и обработкой данных. Встроенные механизмы защиты, такие как предотвращение SQL-инъекций и межсайтовых атак, гарантируют безопасность приложения при работе с конфиденциальной информацией.

.NET MAUI, в свою очередь, является современным решением для создания кросс-платформенных интерфейсов, что особенно важно для реализации приложения, работающего на мобильных устройствах и настольных системах. MAUI позволяет писать общий код для всех платформ, минимизируя затраты на разработку и обслуживание.

Одной из ключевых особенностей .NET MAUI является поддержка адаптивного дизайна, что делает возможным создание интерфейсов, оптимально отображающихся на различных устройствах. Например, приложение для удобного оформления коворкингов будет одинаково удобно выглядеть как на смартфонах, так и на настольных компьютерах. [4]

Дополнительным преимуществом использования ASP.NET Core и .NET MAUI является их интеграция с инструментами Microsoft, такими как Azure DevOps и Visual Studio. Это упрощает процесс разработки, тестирования и развертывания.

Использование этих технологий позволяет создать производительное и гибкое приложение, соответствующее современным требованиям по производительности, безопасности и масштабируемости. Таким образом, ASP.NET Core и .NET MAUI были выбраны как оптимальный технологический стек для разработки приложения для удобного управления коворкингами.

# ТЕОРЕТИЧЕСКОЕ ОБОСНОВАНИЕ РАЗРАБОТКИ ПРОГРАММНОГО ПРОДУКТА

## 3.1 Обоснование необходимости разработки

Создание арендного маркетплейса, объединяющего физические и юридические лица для аренды самых различных объектов, является актуальным проектом, учитывая современные экономические, социальные и технологические тенденции. В последние годы наблюдается рост популярности цифровых платформ, которые упрощают взаимодействие между пользователями и предоставляют им доступ к широкому спектру услуг.

В условиях отсутствия единой площадки для аренды по различным категориям, наш проект призван заполнить этот пробел, предлагая удобное и функциональное решение.

Одной из основных причин разработки маркетплейса является упрощение поиска объектов аренды. Наша платформа решит проблему фрагментации рынка, где сейчас доминируют узконаправленные сервисы для отдельных категорий аренды, таких как недвижимость, автомобили или студийные пространства. Единое решение обеспечит арендаторам возможность экономить время и усилия при поиске, что особенно важно для:

1) пользователей, которым нужно быстро найти разнообразные объекты;

2) малого и среднего бизнеса, нуждающегося в доступе к аренде;

3) жителей удаленных регионов, где выбор ограничен.

Для арендодателей маркетплейс предоставит инструменты для повышения дохода за счет удобного размещения объявлений, автоматизации процессов аренды и взаимодействия с клиентами. Платформа будет стимулировать повышенный спрос, что создаст дополнительную мотивацию для арендодателей активно использовать сервис.

Ключевой экономической выгодой является оптимизация расходов для всех сторон. Арендаторы смогут выбирать наиболее подходящие предложения с учетом цены, условий и местоположения, а арендодатели – снижать затраты на продвижение своих объектов через сторонние каналы.

Таким образом, наш проект станет эффективным решением для упрощения процесса аренды и откроет новые перспективы как для арендаторов, так и для арендодателей. Создание такого маркетплейса является логичным и своевременным шагом на пути к улучшению взаимодействия на рынке аренды и развитию цифровой экономики.

## 3.2 Технологии программирования, используемые для решения

## поставленных задач

Для разработки приложения для бронирования коворкингов были выбраны современные технологии, обеспечивающие высокую производительность, гибкость и удобство кроссплатформенной разработки. Основным языком программирования является C#, а для создания приложения используется платформа .NET MAUI, дополняемая функционалом ASP.NET для серверной части.

**3.2.1** C# – это объектно-ориентированный язык программирования, разработанный корпорацией Microsoft, который используется для создания различных типов приложений, включая веб-приложения, десктопные программы, игры, а также серверные и мобильные приложения. C# является частью платформы .NET и предназначен для того, чтобы объединить мощь и производительность языков, таких как C++ и Java, с простотой и гибкостью современных языков программирования.

История C# начинается с начала 2000-х годов, когда Microsoft искала способ создать язык, который был бы удобен для разработки на платформе .NET. Разработанный под руководством Андерса Хейлсберга, C# стал ключевым компонентом экосистемы .NET, предлагая разработчикам мощный инструмент для создания надежных и производительных приложений.

Одной из ключевых особенностей C# является строгая типизация, что способствует уменьшению количества ошибок на этапе компиляции и упрощает поддержку кода. Это делает его отличным выбором для разработки крупных и сложных систем. C# также поддерживает автоматическое управление памятью благодаря встроенному сборщику мусора, что упрощает работу с памятью и позволяет разработчикам сосредоточиться на логике приложения. Основные преимущества C#:

1. Объектно-ориентированное программирование (ООП): C# – полностью объектно-ориентированный язык, что означает, что все элементы программы (классы, методы, поля и т.д.) представляются в виде объектов. Это упрощает разработку сложных систем и улучшает код через повторное использование, инкапсуляцию и наследование.
2. Интеграция с .NET: Платформа .NET предоставляет множество встроенных библиотек и инструментов, что ускоряет разработку. C# прекрасно интегрирован с .NET, обеспечивая высокую производительность, кросс-платформенность (с появлением .NET Core) и поддержку большого количества сценариев, включая десктопные приложения (WPF, Windows Forms), веб-разработку (ASP.NET), облачные сервисы, игры (Unity) и мобильные приложения (Xamarin, .NET MAUI).
3. Поддержка асинхронного программирования: C# предоставляет встроенную поддержку асинхронного программирования через ключевые слова async и await. Это позволяет создавать высокопроизводительные приложения, которые могут работать с асинхронными операциями, такими как запросы к базе данных или обращения к внешним сервисам, не блокируя основной поток выполнения.
4. LINQ (Language Integrated Query): Одно из самых значительных нововведений C# – это LINQ, который позволяет писать SQL-подобные запросы к коллекциям данных (например, к массивам, спискам, базам данных) прямо на уровне языка, делая работу с данными более естественной и простой.
5. Безопасность типов и управление памятью: C# имеет строгую систему типов, которая помогает предотвращать многие распространенные ошибки на уровне компиляции. Также язык поддерживает автоматическое управление памятью через сборщик мусора (Garbage Collector), что снижает вероятность утечек памяти и упрощает работу с памятью.
6. Кросс-платформенность: С появлением .NET Core (и позднее .NET 5 и выше) C# стал кросс-платформенным. Теперь можно разрабатывать приложения не только под Windows, но и под macOS и Linux, что значительно расширяет сферу применения языка.
7. Широкая экосистема: C# поддерживается мощной экосистемой инструментов разработки. Среди них – Visual Studio и Visual Studio Code, которые обеспечивают отличные возможности отладки, тестирования и улучшения производительности.
8. Многопарадигмальный подход: Помимо объектно-ориентированного программирования, C# поддерживает функциональное программирование, позволяя разработчикам комбинировать разные подходы в зависимости от задач.
9. Современные возможности: C# активно развивается и добавляет современные функции, такие как нулевые типы ссылок (для предотвращения ошибок с null), диспетчеризация на основе шаблонов (pattern matching), поддержка асинхронных потоков и другие инструменты для написания более эффективного и читабельного кода.
10. Сообщество и поддержка: C# имеет огромное сообщество разработчиков, множество обучающих материалов, форумов и библиотек, что облегчает обучение и поддержку проектов. Microsoft также активно поддерживает язык, регулярно выпуская обновления и добавляя новые возможности.
11. Веб-разработка: ASP.NET Core позволяет разрабатывать масштабируемые веб-приложения и API с высокой производительностью.
12. Игры: C# используется в Unity – одном из самых популярных игровых движков.
13. Мобильные приложения: С помощью Xamarin и .NET MAUI можно разрабатывать кроссплатформенные мобильные приложения для Android и iOS.
14. Корпоративные системы: C# широко используется для создания сложных корпоративных решений, таких как ERP-системы, CRM и другие.

## 3.2.2 NET MAUI является открытым исходным кодом и является эволюцией Xamarin.Forms, расширенной от мобильных до классических сценариев, с элементами управления пользовательским интерфейсом, перестроенными с нуля для повышения производительности и расширяемости. Если вы ранее использовали Xamarin.Forms для создания кроссплатформенных пользовательских интерфейсов, вы заметите множество сходств с .NET MAUI. Однако между ними также есть некоторые различия. С помощью .NET MAUI можно создавать приложения для нескольких платформ в одном проекте и при необходимости добавлять исходный код и ресурсы для конкретной платформы. Одной из основных целей .NET MAUI является возможность реализовать в общей базе кода как можно больше логики и макета пользовательского интерфейса для приложения [4].

.NET Multi-platform App UI (.NET MAUI) – это эволюция Xamarin.Forms, которая расширяет возможности за пределы мобильных Android и iOS в настольные приложения для Windows и macOS. С .NET MAUI вы можете создавать приложения, которые отлично работают на любом устройстве под управлением Windows, macOS, Android и iOS из единой кодовой базы. В сочетании с инструментами повышения производительности и эмуляторами Visual Studio, .NET и Visual Studio значительно ускоряют процесс разработки для создания приложений, ориентированных на максимально широкий набор устройств. Используйте единый стек разработки, который поддерживает лучшие в своем классе решения для всех современных рабочих нагрузок с унифицированным SDK, библиотеками базовых классов и цепочкой инструментов [5].

Первая версия MAUI вышла в мае 2022 года, далее в ноябре 2023 года вышли обновления для .NET 8. Однако стоит отметить, что сам фреймворк .NET Multi-platform App UI фактически представляет эволюцию другого фреймворка - Xamarin Forms, который имеет более чем 10-летнюю историю и который с 2016 года был куплен и развивался компанией Microsoft. Поэтому, имя опыт работы с Xamarin Forms, не составит больших проблем перейти на .NET MAUI.

.NET MAUI предназначен для разработчиков, которые нуждаются в:

– Написание кроссплатформенных приложений в XAML и C# из одной общей базы кода в Visual Studio;

– Совместное использование макета пользовательского интерфейса и разработка на различных платформах;

– Совместное использование кода, тестов и бизнес-логики на разных платформах.

.NET MAUI объединяет API-интерфейсы Android, iOS, macOS и Windows в единый API, который предоставляет разработчикам возможность выполнять однократную запись в любом месте, обеспечивая дополнительный доступ к каждому аспекту каждой собственной платформы.

.NET 6 или более поздней версии предоставляет ряд платформ для создания приложений: .NET для Android, .NET для iOS, .NET для Mac Catalyst и библиотеки Пользовательского интерфейса Windows 3 (WinUI 3). Все эти платформы имеют доступ к одной библиотеке базовых классов .NET (BCL). Эта библиотека абстрагирует сведения о базовой платформе от кода. Библиотека BCL имеет зависимость от среды выполнения .NET, в которой выполняется код приложения. Для Android, iOS и macOS среда реализуется Mono, реализация среды выполнения .NET. В Windows .NET CoreCLR предоставляет среду выполнения.

.NET MAUI предоставляет единый фреймворк для создания приложений. Однако в процессе работы он опирается на ряд субплатформ, через которые идет взаимодействие с каждой отдельной операционной системой: .NET for Android, .NET for iOS, .NET for macOS и Windows UI 3 (WinUI 3). И в общем случае код приложения сначала обращается к платформе .NET MAUI, а та затем обращается к субплатформе для конкретной ОС. Хотя фреймворк также позволяет напрямую обращаться коду приложения к этим субплатформам.

Хотя BCL позволяет приложениям, работающим на разных платформах, совместно использовать общую бизнес-логику, различные платформы имеют различные способы определения пользовательского интерфейса для приложения, а также предоставляют различные модели для указания того, как элементы пользовательского интерфейса взаимодействуют и взаимодействуют. Вы можете создать пользовательский интерфейс для каждой платформы отдельно с помощью соответствующей платформы (.NET для Android, .NET для iOS, .NET для Mac Catalyst или WinUI 3), но этот подход требует поддержания базы кода для каждого отдельного семейства устройств.

.NET MAUI предоставляет единую платформу для создания пользовательских интерфейсов для мобильных и классических приложений. Высокоуровневое представление архитектуры приложения .NET MAUI показано на рисунке 1.

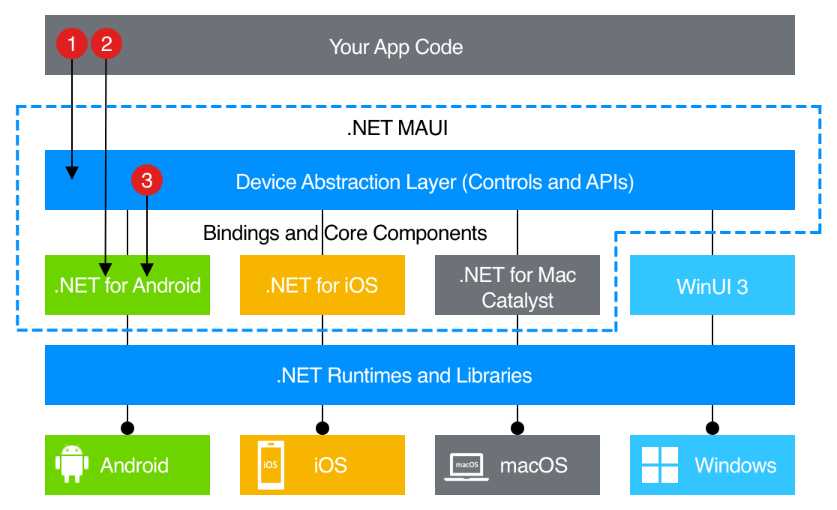


Рисунок 1 – Высокоуровневое представление

архитектуры приложения .NET MAUI

В приложении .NET MAUI пишется код, который в основном взаимодействует с элементами управления .NET MAUI и уровнем API. Затем этот слой напрямую использует собственные API платформы. Кроме того, код приложения может напрямую выполнять API платформы, если это необходимо.

Приложения .NET MAUI можно записывать на ПК или Mac и компилировать в собственные пакеты приложений:

Приложения Android, созданные с помощью .NET MAUI, компилируются из C# на промежуточный язык (IL), который затем компилируется в собственную сборку при запуске приложения.

Приложения iOS, созданные с помощью .NET MAUI, полностью заранее компилируются из C# в собственный код сборки ARM.

Приложения macOS, созданные с помощью .NET MAUI, используют Mac Catalyst, решение от Apple, которое приносит ваше приложение iOS, созданное с помощью UIKit на рабочий стол, и расширяет его с помощью дополнительных API AppKit и платформы по мере необходимости.

Приложения Windows, созданные с помощью .NET MAUI, используют библиотеку Пользовательского интерфейса Windows 3 (WinUI 3) для создания собственных приложений, предназначенных для рабочего стола Windows.

.NET MAUI предоставляет коллекцию элементов управления, которые можно использовать для отображения данных, запуска действий, указания действий, отображения коллекций, выбора данных и т. д. Помимо коллекции элементов управления, .NET MAUI также предоставляет следующее:

– Продуманный механизм визуализации для проектирования страниц. Несколько типов страницы для создания полнофункциональных типов навигации, таких как панели;

– Поддержка привязки данных для создания более элегантного и удобного в обслуживании кода;

– Возможность настраивать обработчики для улучшения способа представления элементов пользовательского интерфейс;

– Кроссплатформенные API для доступа к собственным функциям устройств. Эти API позволяют приложениям получать доступ к функциям устройств, таким как GPS, акселерометр и состояния батареи и сети;

– Кроссплатформенная функциональность графики, которая предоставляет холст рисования, который поддерживает рисование и рисование фигур и изображений, операций создания и преобразования графических объектов;

– Единая система проектов, использующая многоцелевой объект для целевой платформы Android, iOS, macOS и Windows;

– Горячая перезагрузка .NET, чтобы можно было изменить xaml и управляемый исходный код во время работы приложения, а затем наблюдать за результатом изменений без перестроения приложения.

Единый проект .NET MAUI принимает возможности разработки для конкретной платформы, которые обычно возникают при разработке приложений и абстрагирует их в один общий проект, который может нацелен на Android, iOS, macOS и Windows.

Единый проект .NET MAUI обеспечивает упрощенную и согласованную кроссплатформенную разработку независимо от целевых платформ. Единый проект .NET MAUI предоставляет следующие возможности:

– Один общий проект, предназначенный для Android, iOS, macOS и Windows;

– Упрощенный выбор целевого объекта отладки для запуска приложений .NET MAUI;

– Общие файлы ресурсов в одном проекте;

– Один манифест приложения, указывающий название, идентификатор и версию приложения;

– Доступ к api-интерфейсам и средствам для конкретной платформы при необходимости;

– Одна кроссплатформенная точка входа приложения.

Единый проект .NET MAUI включен с помощью многоцелевой и использования проектов в стиле SDK.

.NET MAUI включает поддержку горячей перезагрузки .NET, которая позволяет изменять управляемый исходный код во время работы приложения без необходимости вручную приостановить или попасть в точку останова. Затем изменения кода можно применить к работающему приложению без перекомпиляции.

**3.2.3** SP.NET Core – это современная технология для создания веб-приложений на платформе .NET, разработанная Microsoft. Для работы с этой технологией используются языки программирования C# и F#, что обеспечивает удобство и мощные возможности для разработчиков.

История ASP.NET началась в 2002 году с выходом первой версии платформы .NET, после чего обе технологии развивались синхронно. Изначально ASP.NET функционировала только в среде Windows с использованием веб-сервера IIS. Однако с 2014 года произошли значительные изменения: Microsoft сделала ASP.NET открытой и кроссплатформенной технологией, что стало настоящим прорывом в её развитии.

Можно выделить несколько основных преимуществ ASP.NET [6]:

1 Производительность. Код приложения или страницы, созданных с помощью этой платформы, компилируется по необходимости. Кроме того, компиляция осуществляется только один раз, а машинный код сохраняется в системные папки.

2 Кроссплатформенность. Хотя ASP.NET разрабатывается и поддерживается Microsoft, является собственностью этой корпорации, она доступна для использования на ПК с операционными системами MacOS и Linux. Это существенно расширяет число программистов, которые могут использовать эту платформу для создания веб-приложений.

3 Простота разработки. ASP.NET создана в рамках основного подхода Microsoft к программированию как к максимально простому процессу, многие функции и стадии которого можно автоматизировать.

**3.2.4** Dapper – библиотека для реализации взаимодействия с базой данных с высокой производительностью, простотой использования и гибкостью при работе с SQL-запросами. Эта библиотека предоставляет разработчику удобный инструментарий для быстрого и эффективного выполнения запросов, что делает её отличным выбором для проектов с акцентом на прямое управление данными.

Одним из ключевых преимуществ Dapper является его высокая скорость. Благодаря минимальному количеству дополнительной логики при обработке запросов, Dapper работает почти на том же уровне производительности, что и использование чистого ADO.NET. Это позволяет приложению эффективно обрабатывать большие объемы данных, что особенно актуально для задач с интенсивным использованием базы данных.

Гибкость Dapper выражается в его подходе к работе с SQL-запросами. Разработчик имеет полный контроль над выполняемыми запросами, что упрощает написание сложной логики взаимодействия с базой данных. Это обеспечивает точное соответствие запросов бизнес-логике приложения и позволяет избежать накладных расходов, характерных для более высокоуровневых инструментов.

Дополнительно, Dapper упрощает маппинг результатов запросов на объекты в коде. Библиотека автоматически связывает столбцы результата запроса с свойствами объекта, что существенно ускоряет процесс разработки. Возможность легко связывать данные с объектами также делает код более читаемым и поддерживаемым. [7]

# ПРОЕКТИРОВАНИЕ ФУНКЦИОНАЛЬНЫХ ВОЗМОЖНОСТЕЙ ПРОГРАММЫ

**4.1 Настройка взаимодействия сервера и базы данных**

Приложение построено на принципах **чистой архитектуры**, обеспечивающей четкое разделение ответственности между слоями. Такая структура способствует масштабируемости, упрощает тестирование и поддержку кода, а также делает приложение устойчивым к изменениям технологий.

В рамках слоя **инфраструктуры** реализованы ключевые компоненты для работы с базой данных: **репозитории** и **единица работы** (Unit of Work). Эти компоненты обеспечивают удобный и безопасный доступ к данным приложения. Для взаимодействия с базой данных используется библиотека **Dapper**, которая предоставляет возможности для выполнения чистых SQL-запросов, сохраняя их прозрачность и управляемость. [8]

Каждый репозиторий отвечает за выполнение специфических операций с данными, относящимися к определенной сущности. Например, репозиторий объявлений предоставляет методы для получения списка доступных объявлений, их создания и объявления. Методы репозиториев реализуют SQL-запросы, которые напрямую взаимодействуют с базой данных, обеспечивая высокую производительность и контроль над запросами.

**Единица работы (Unit of Work)** координирует выполнение транзакций между несколькими репозиториями, гарантируя согласованность данных. Это особенно важно для операций, затрагивающих несколько таблиц, таких как бронирование коворкинга и обновление статуса пользователя в рамках одной транзакции.

На уровне **бизнес-логики** (слой Application) используются методы репозиториев для выполнения конкретных сценариев приложения. Каждый Use Case представляет собой самостоятельный блок, инкапсулирующий логику выполнения операции.

Слой **ASP.NET Web API контроллеров** отвечает за взаимодействие с клиентской частью приложения. Контроллеры принимают HTTP-запросы от пользователей, преобразуют их в параметры для соответствующего Use Case, а затем передают результат обратно клиенту в виде ответа API.

Также на уровне сервера реализовано хранилище изображений. Основная часть этого хранилища – папка с файлами, расположенная в wwwroot прокта. Это позволяет получить доступ к картинкам по url.

**4.2 Настройка взаимодействия клиента и сервера**

Клиентская и серверная части приложения взаимодействуют через четко определенные контракты, основанные на DTO, которые определены в слое **Application.**

Как клиент, так и сервер работают исключительно с DTO, которые определены в слое **Application.** Эти объекты представляют собой упрощенные модели данных, содержащие только те свойства, которые необходимы для конкретной операции. Это упрощает обмен данными и обеспечивает единообразие структуры запросов и ответов.

Клиент ничего не знает о конечных точках API. Вся логика работы с конечными точками, включая их вызовы, параметры и обработку ответов, сосредоточена в **сервисах клиентского приложения.** Это позволяет легко изменять серверное API без необходимости модификации клиентского кода.

Вся бизнес-логика сосредоточена на сервере. Клиент выполняет роль интерфейса, отображающего данные и обрабатывающего пользовательский ввод. Это повышает безопасность, предотвращая выполнение критически важной логики на стороне клиента.

На клиенте реализован слой презентации с использованием сервисов, которые:

1. Вызывают нужные конечные точки сервера через HTTP-запросы;

2. Формируют запросы на основе предоставленных DTO;

3. Обрабатывают ответы сервера, преобразуя их в объекты.

На серверной стороне структура взаимодействия состоит из следующих шагов:

Контроллеры принимают HTTP-запросы преобразуют данные запроса в DTO и передают их в соответствующий Use Case. Контроллеры не содержат бизнес-логики, что делает их легкими для тестирования и поддержки.

После выполнения бизнес-логики Use Case возвращает DTO с данными, которые контроллер преобразует в HTTP-ответ для клиента.

# ПРОЕКТИРОВАНИЕ РАЗРАБАТЫВАЕМОЙ БАЗЫ ДАННЫХ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

# ЗАКЛЮЧЕНИЕ

# СПИСОК ЛИТЕРАТУРНЫХ ИСТОЧНИКОВ

[1] PostgreSQL https://blog.skillfactory.ru/glossary/postgresql/

[2] История PostgreSQL https://www.inp.nsk.su/~baldin/PostgreSQL/postgresql-intro.pdf

[3] Безопасность Windows https://support.hp.com/kz-ru/document/ish\_3187108-3103836-16

[4] Maui open source [Электронный ресурс]. – Режим доступа: https://github.com/dotnet/maui.

[5] .NET MAUI [Электронный ресурс]. – Режим доступа: https://learn.microsoft.com/ru-ru/dotnet/maui/what-is-maui?view=net-maui-8.0 .

[6] Технология ASP.NET [Электронный ресурс]. – Режим доступа: https://blog.skillfactory.ru/glossary/asp-net/ – Дата доступа: 29.10.2023.

[7] Dapper на примере https://habr.com/ru/articles/665836/

[8] Чистая архитектура на примере https://habr.com/ru/articles/784922/

# ПРИЛОЖЕНИЕ А

**(обязательное)**

**Листинг программного кода**

# ПРИЛОЖЕНИЕ Б

**(обязательное)**

**Конечная схема базы данных**

# ПРИЛОЖЕНИЕ В

**(обязательное)**

**Ведомость курсового проекта**

# ПРИЛОЖЕНИЕ Г

**Графический интерфейс**