Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет   
информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина «Операционные среды и системное программирование»

|  |  |
| --- | --- |
|  | *К защите допустить:* |
|  | И. о. заведующего кафедрой информатики  *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_* С. И. Сиротко |
|  |  |

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

на тему

**ПРОГРАММНОЕ СРЕДСТВО ДЛЯ ВЕДЕНИЯ ЛИЧНОЙ БУХГАЛТЕРИИ**

БГУИР КП 1-40 04 01 005 ПЗ

Студент          С. А. Брыкульский

Руководитель          А. В. Давыдчик

Нормоконтролер          А. В. Давыдчик

Минск 2024

# СОДЕРЖАНИЕ

[СОДЕРЖАНИЕ 2](#_Toc183349963)

[ВВЕДЕНИЕ 3](#_Toc183349964)

[1 ТЕОРИТИЧЕСКОЕ ОБОСНОВАНИЕ РАЗРАБОТКИ 4](#_Toc183349965)

[1.1 Обоснование необходимости разработки 4](#_Toc183349966)

[1.1.1 Актуальность управления личными финансами 4](#_Toc183349967)

[1.1.2. Анализ существующих решений 4](#_Toc183349968)

[1.1.3. Проблемы существующих решений 5](#_Toc183349969)

[1.1.4. Цели разработки 5](#_Toc183349970)

[1.2 Технологии программирования, используемые для решения поставленных задач 5](#_Toc183349971)

[1.2.1 PostgreSQL 5](#_Toc183349972)

[2 ПРОЕКТИРОВАНИЕ ФУНКЦИОНАЛЬНЫХ ВОЗМОЖНОСТЕЙ ПРОГРАММЫ 7](#_Toc183349973)

[3 ПРОЕКТИРОВАНИЕ РАЗРАБАТЫВАЕМОЙ БАЗЫ ДАННЫХ РАЗРАБАТЫВАЕМОГО ПРИЛОЖЕНИЯ 8](#_Toc183349974)

[4 РАЗРАБОТКА БАЗЫ ДАННЫХ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ 9](#_Toc183349975)

# ВВЕДЕНИЕ

В настоящее время ведение личной бухгалтерии и финансового учета становится важным аспектом управления личными финансами. Это особенно актуально для людей, стремящихся контролировать свои доходы и расходы, планировать бюджет, а также оптимизировать использование денежных средств. Наличие инструментов для упрощения таких процессов помогает пользователям систематизировать финансовую информацию, избегать ошибок и принимать более взвешенные решения.

Целью данной курсовой работы является разработка программного средства для ведения личной бухгалтерии, которое будет адаптировано для повседневных нужд пользователей. Программа должна предоставлять функциональность для управления личными финансами, учета доходов и расходов, а также анализа финансовой активности.

Для выполнения целей курсовой работы необходимо выполнить следующие задачи:

1. Анализ существующих программных решений для ведения личной бухгалтерии с целью изучения их функциональности.
2. Проектирование структуры базы данных подходящей для хранения информации о доходах, расходах, категориях транзакций и других финансовых данных.
3. Реализация базы данных, обеспечивающей эффективное хранение и обработку данных.
4. Разработка веб-приложения для взаимодействия конечного пользователя с базой данных, включая функциональность учета финансов и анализа данных.
5. Проектирование пользовательского интерфейса для обеспечения удобного доступа к функциям приложения.

Данный курсовой проект оформлен с помощью СТП 01-2017 (стандарт предприятия) [1].

# 1 ТЕОРИТИЧЕСКОЕ ОБОСНОВАНИЕ РАЗРАБОТКИ

## 1.1 Обоснование необходимости разработки

## 1.1.1 Актуальность управления личными финансами

Современный мир характеризуется высоким уровнем экономической активности и финансовой осведомленности среди населения. Финансовая грамотность становится неотъемлемой частью жизни, поскольку управление личными финансами позволяет не только контролировать текущие доходы и расходы, но и эффективно планировать будущее. Современные финансы становятся преимущественно цифровыми. Большинство операций – от оплаты товаров и услуг до перевода средств происходит в электронном формате. Это обусловлено развитием интернет-банкинга, мобильных платежных систем и ростом популярности безналичных расчетов.

Однако цифровизация финансов предъявляет новые требования к пользователям. Для эффективного управления личными финансами требуется инструмент, который позволит не только учитывать все финансовые операции, но и предоставит аналитику, визуализацию данных. Это особенно важно в условиях:

1. Увеличения количества источников доходов и расходов (банковские счета, электронные кошельки, криптовалютные кошельки).
2. Необходимости планирования крупных трат или сбережений.
3. Роста объема данных, требующих систематизации и анализа.

## 1.1.2. Анализ существующих решений

На рынке представлено множество решений для учета личных финансов, большая часть из которых ориентирована на мобильные устройства. Эти приложения предоставляют пользователям возможность учитывать транзакции, формировать отчеты и ставить финансовые цели. Однако у большинства из них есть существенные ограничения:

1. Мобильные приложения (например, CoinKeeper, Money Manager):

* Предлагают ограниченный функционал в бесплатных версиях.
* Могут быть избыточными по количеству функций или слишком сложными для пользователей, которым нужен базовый учет.

1. Табличные процессоры (Microsoft Excel, Google Sheets):

* Требуют значительных навыков для настройки формул, макросов и визуализаций.
* Не всегда удобны для регулярного использования из-за отсутствия автоматизации.
* Не предлагают готового интерфейса или инструментов для анализа данных.

1. Интернет-банкинг:

* Ограничивается учетами транзакций только в рамках одного банка.
* Не поддерживает учет наличных средств или данных из сторонних источников.

## 1.1.3. Проблемы существующих решений

Большинство существующих решений сталкиваются с рядом проблем:

1. Отсутствие комплексного подхода к учету всех финансовых операций пользователя.
2. Высокий порог входа для настройки (в случае использования Excel или Google Sheets).
3. Недостаточная гибкость для персонализации интерфейса и отчетности.
4. Зависимость от наличия интернета и невозможность работы с данными в оффлайн-режиме.

Эти ограничения создают спрос на программное средство, которое сможет объединить простоту использования, широкий функционал и поддержку как мобильных, так и десктопных пользователей.

## 1.1.4. Цели разработки

Цели создания нового программного средства в рамках курсовой работы включают:

1. Обеспечение простого и интуитивного интерфейса, доступного для пользователей с любым уровнем финансовой грамотности.
2. Внедрение функций учета доходов и расходов, анализа финансов и составления отчетов.
3. Реализация механизма визуализации данных для упрощения анализа финансовой информации.

## 1.2 Технологии программирования, используемые для решения поставленных задач

## 1.2.1 PostgreSQL

PostgreSQL — это реляционная система управления базами данных (СУБД), известная своей надежностью, масштабируемостью и богатым функционалом. Ее история начинается в середине 1980-х годов, когда в Калифорнийском университете в Беркли началась разработка проекта POSTGRES под руководством Майкла Стоунбрейкера. Первоначальной целью проекта было создание системы, способной работать с расширенными типами данных, включая сложные объекты и их отношения. Впоследствии, в 1996 году, проект был переименован в PostgreSQL для обозначения его соответствия стандарту SQL.

Одной из ключевых особенностей PostgreSQL является ее ориентация на открытость и расширяемость. Система имеет открытый исходный код, что позволяет разработчикам по всему миру вносить улучшения, создавать расширения и адаптировать систему для специфических задач. Это делает PostgreSQL универсальным инструментом, подходящим как для небольших приложений, так и для крупных корпоративных систем с высокими требованиями к надежности и производительности.

Важным этапом в развитии PostgreSQL стало внедрение поддержки ACID-транзакций, что обеспечило гарантии сохранности данных даже в условиях сбоя системы. Другим значительным достижением стала интеграция работы с полуструктурированными данными, включая JSON и JSONB, что позволило использовать PostgreSQL в проектах, где требуется гибкость в работе с данными. Благодаря этим возможностям, PostgreSQL стала популярной платформой для финансовых систем, систем управления контентом, аналитических приложений и других сфер, где требуется высокая точность обработки данных.

Современные версии PostgreSQL обладают широким набором функциональных возможностей. Среди них можно выделить поддержку индексации, включая полнотекстовый поиск, оптимизацию запросов с использованием планировщиков и работу с большим числом параллельных транзакций. Эти функции делают PostgreSQL особенно эффективной для приложений, где важна производительность, включая системы аналитики и обработки больших данных.

Еще одним важным преимуществом является возможность работы с пользовательскими типами данных. Например, разработчик может создавать собственные структуры, которые интегрируются в базу данных так же, как и стандартные типы, такие как числовые или строковые. Это позволяет использовать PostgreSQL в нестандартных сценариях, где требуется работа с географическими координатами, временными рядами или другими специфическими данными.

Выбор PostgreSQL обусловлен ее надежностью, расширяемостью и высокой степенью соответствия современным требованиям разработки. Ее поддержка стандарта SQL обеспечивает совместимость с широким спектром приложений и инструментов, а открытый исходный код позволяет легко интегрировать систему с другими компонентами веб-приложения.

PostgreSQL успешно применяется в самых разных областях, включая электронную коммерцию, аналитику, управление финансами и социальные сети. В каждом из этих случаев ее гибкость, поддержка различных типов данных и способность масштабироваться под растущие нагрузки играют ключевую роль. Благодаря этим характеристикам PostgreSQL занимает одну из ведущих позиций среди современных СУБД и продолжает активно развиваться, предоставляя пользователям новые возможности с каждой версией.

## 1.2.2 C# и ASP.NET Web API

C# ­ это современный объектно-ориентированный язык программирования, разработанный компанией Microsoft. Он был создан как часть платформы .NET и изначально задумывался для разработки корпоративных приложений и веб-решений. Сегодня C# является одним из наиболее популярных языков программирования благодаря своей универсальности, строгой типизации и обширной экосистеме. В сочетании с ASP.NET Web API, который представляет собой платформу для построения RESTful веб-сервисов, C# становится мощным инструментом для разработки серверной части веб-приложений.

Язык C# выделяется своей строгой типизацией, что способствует предотвращению множества ошибок еще на этапе компиляции. Это особенно важно при работе с финансовыми приложениями, где требуется высокая точность обработки данных. Одной из ключевых особенностей языка является его гибкость: он поддерживает множество парадигм программирования, включая объектно-ориентированное, функциональное и асинхронное программирование. Последнее играет важную роль при создании высокопроизводительных веб-приложений, где важно обрабатывать запросы пользователей параллельно и без блокировок.

ASP.NET Web API ­ это технология, входящая в состав платформы ASP.NET, предназначенная для создания веб-сервисов, которые взаимодействуют с клиентскими приложениями по протоколу HTTP. Она позволяет разрабатывать серверную часть приложений, которая может обслуживать запросы из самых разных источников, включая веб-браузеры, мобильные приложения и другие серверы. Одной из ключевых особенностей ASP.NET Web API является его ориентация на принципы REST (Representational State Transfer), что делает эту платформу идеальным выбором для создания распределенных систем.

ASP.NET Web API построен на основе MVC-архитектуры, которая обеспечивает четкое разделение логики приложения на три компонента: модели, представления и контроллеры. Это упрощает разработку и тестирование приложения, а также повышает его поддерживаемость. Разработчики могут легко организовать обработку запросов, манипуляцию данными и передачу ответов в формате JSON или XML. Это особенно важно для современных веб-приложений, где требуется высокая скорость передачи данных и поддержка разных форматов.

Преимущества использования C# и ASP.NET Web API в разработке серверной части данного приложения многочисленны. Во-первых, это интеграция с платформой .NET, которая предоставляет множество библиотек и инструментов для упрощения разработки. Например, библиотека Entity Framework позволяет работать с базой данных на уровне объектов, что ускоряет процесс разработки и снижает вероятность ошибок. Во-вторых, это поддержка современных стандартов разработки, включая асинхронные операции, что повышает производительность приложения. Наконец, это богатый инструментарий для тестирования и отладки, который позволяет разработчикам создавать надежные и безопасные приложения.

Стоит отметить и безопасность, которая является важным аспектом разработки веб-приложений. ASP.NET Web API предоставляет встроенные механизмы для защиты данных, включая аутентификацию, авторизацию и шифрование. Эти механизмы позволяют защитить приложение от большинства распространенных угроз, таких как SQL-инъекции, межсайтовый скриптинг и атаки типа "человек посередине".

C# и ASP.NET Web API также обладают широкой экосистемой и сильной поддержкой со стороны сообщества разработчиков. Это означает, что разработчики могут найти множество ресурсов для обучения, решения технических проблем и обмена опытом. Платформа постоянно обновляется, предлагая новые функции и улучшения, что делает ее актуальным выбором для разработки современных приложений.

Особое внимание следует уделить интеграции C# и ASP.NET Web API с другими технологиями. Например, в рамках данного проекта эти инструменты взаимодействуют с PostgreSQL через ORM-библиотеку Entity Framework. Это позволяет значительно упростить доступ к данным, абстрагируя разработчиков от работы с SQL-запросами. Вместо этого они могут работать с данными на уровне объектов, что делает код более читабельным и простым для сопровождения.

Еще одним важным преимуществом ASP.NET Web API является его масштабируемость. Эта технология позволяет легко адаптировать приложение под увеличение нагрузки, добавляя новые серверы или перераспределяя ресурсы. Это особенно важно для финансовых приложений, где число пользователей и объем данных может значительно возрасти со временем.

Таким образом, выбор C# и ASP.NET Web API для реализации серверной части данного проекта обусловлен их мощным функционалом, надежностью и широкими возможностями для интеграции с другими компонентами. Эти технологии предоставляют разработчикам все необходимые инструменты для создания высококачественного программного обеспечения, которое отвечает современным требованиям к производительности, безопасности и удобству использования.

## 1.2.3. Next.js

Next.js ­ это прогрессивный фреймворк для разработки веб-приложений, построенный поверх библиотеки React и предоставляющий мощные инструменты для создания интерактивных пользовательских интерфейсов. Разработанный компанией Vercel, Next.js представляет собой удобный и гибкий инструмент, который упрощает многие аспекты фронтенд-разработки, такие как маршрутизация, управление рендерингом страниц и интеграция с различными API. Он зарекомендовал себя как идеальное решение для приложений, где важны производительность, SEO-оптимизация и удобство разработки.

Одной из ключевых характеристик Next.js является поддержка серверного рендеринга (Server-Side Rendering, SSR), который позволяет генерировать HTML-код страниц на сервере перед их отправкой клиенту. Этот подход значительно улучшает производительность приложений, так как сокращает время первой загрузки страницы и делает ее доступной для индексации поисковыми системами. Кроме SSR, Next.js поддерживает статическую генерацию страниц (Static Site Generation, SSG), при которой HTML-страницы создаются заранее на этапе сборки приложения. Это обеспечивает почти мгновенную загрузку страниц пользователями, что особенно важно для приложений с высокой посещаемостью.

Маршрутизация в Next.js реализована через файловую систему, что делает процесс создания маршрутов простым и понятным. Структура папки pages автоматически преобразуется в маршруты приложения, позволяя разработчикам сосредоточиться на логике приложения вместо настройки сложных конфигураций. Для более сложных сценариев, таких как динамическая маршрутизация, предусмотрены гибкие механизмы управления URL, которые дают возможность создавать масштабируемые и адаптивные приложения.

Следует отметить, что Next.js включает встроенную поддержку TypeScript ­ языка, который расширяет возможности JavaScript за счет введения строгой типизации. Использование TypeScript в проектах на Next.js делает разработку более безопасной, так как ошибки обнаруживаются еще на этапе написания кода. Это особенно важно для сложных веб-приложений, где требуется высокая надежность и читаемость кода.

Важной составляющей Next.js является API-роутинг, который позволяет создавать серверные функции непосредственно в структуре фронтенд-приложения. Это упрощает реализацию взаимодействия между клиентской частью и серверной логикой, особенно в проектах, где серверная часть не требует масштабных решений. Кроме того, встроенные инструменты оптимизации производительности, такие как автоматическая оптимизация изображений и динамическое разделение кода, помогают минимизировать объем передаваемых данных и ускоряют работу приложения.

В рамках данного проекта Next.js используется как основной инструмент для создания пользовательского интерфейса. Фреймворк обеспечивает удобную интеграцию с серверной частью, реализованной на ASP.NET Web API, и позволяет эффективно использовать возможности серверного рендеринга и статической генерации. Преимущество использования Next.js заключается в его способности сочетать современные технологии фронтенд-разработки с высокой производительностью, удобством работы и простотой интеграции, что делает его выбором не только для данного проекта, но и для разработки современных веб-приложений в целом.

# 2 ПРОЕКТИРОВАНИЕ ФУНКЦИОНАЛЬНЫХ ВОЗМОЖНОСТЕЙ ПРОГРАММЫ

# 3 ПРОЕКТИРОВАНИЕ РАЗРАБАТЫВАЕМОЙ БАЗЫ ДАННЫХ РАЗРАБАТЫВАЕМОГО ПРИЛОЖЕНИЯ

# 4 РАЗРАБОТКА БАЗЫ ДАННЫХ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ