## Аннотация

В пояснительной записке описывается Web API для управления личными финансами, реализованного на базе ASP.NET Core. Приложение предоставляет пользователям возможность учета расходов, установки бюджетов и анализа финансовых привычек. В записке представлены архитектура системы, описание основных компонентов API, а также примеры использования.

## 

**Содержание**

## Введение……………………………………………………………………………...4

1 Назначение и область применения…………………………………………..….5

2 Постановка задачи…………………………………………………………..…....6

3 Описание программы………………………………………………………….....7

3.1 Общие сведения……………………………………………………...........7

3.2 Функциональное назначение………………………………..………..…..8

3.3 Описание логической структуры………………………………….....…..9

3.4 Используемые технические средства…………………………….……..10

3.5 Вызов и загрузка……………………………………………………….....11

3.6 Входные и выходные данные……………………………………..….….12

4 Программа и методика испытаний…………………………………………..…97

4.1 Объекты испытаний……………………………………………..…….…97

4.2 Цель испытаний……………………………………………………...…..97

4.3 Требования к программе…………………………………………...……97

4.4 Методы испытаний…………………………………………………........97

4.5 Тестовый пример……………………………………………………....…98

5 Руководство оператора………………………………………………………...100

5.1 Выполнение программы…………………………………………….…100

5.2 Сообщение оператору……………………………………………….....102

Заключение……………………………………………………………………...…123

Приложения. Текст программы ...………………………………………………..124

Источники, использованные при разработке…………………………………....129

## Введение

Web API, предназначено для предоставления пользователям возможности отслеживать свои расходы, устанавливать финансовые цели и получать аналитические данные о своих финансовых привычках. Разработка Web API осуществлялась на основании:

1. Курсовое задание;
2. Техническое задание (при наличии).

**1 Назначение и область применения**

Назначением разработанного Web API является предоставление программного интерфейса для приложений, позволяющих пользователям эффективно управлять своими личными финансами. Web API предоставляет функциональность для учета расходов, планирования бюджетов и анализа финансовых привычек. Область применения Web API – мобильные и веб-приложения для персонального финансового менеджмента.

Существующие аналоги Web API:

* CoinKeeper (https://about.coinkeeper.me/) – Мобильное приложение для отслеживания трат и аналитики.
* Cashew (https://cashewapp.web.app/) – Мобильное приложение для отслеживания трат и аналитики.

**2 Постановка задачи**

Необходимо разработать Web API для системы управления личными финансами, обеспечивающего пользователям возможность учета расходов, планирования бюджетов и анализа финансового состояния. Разработанный API должен соответствовать следующим требованиям:

* Предоставление функциональности CRUD (Create, Read, Update, Delete) для именованных денежных хранилищ.
* Предоставление функциональности CRUD для именованных категорий расходов.
* Реализация функциональности добавления, получения и удаления транзакций.
* Обеспечение авторизации и контроля доступа, гарантирующего привязку данных к пользователю и исключающего доступ других пользователей.
* Реализация регистрации пользователей с валидацией данных (ФИО, email, пароль, соответствующий требованиям безопасности: длина, спецсимволы, заглавные буквы).

API должен обеспечивать надежность работы посредством:

* Корректной обработки ошибок при запросе несуществующих данных.
* Предотвращения добавления дубликатов данных.
* Валидации входных данных (формат, диапазоны, логическая корректность) и отклонения запросов при обнаружении ошибок.

Входные и выходные данные представлены в формате JSON с кодировкой UTF-8.

Применяемые методы: При разработке API будут использоваться стандартные методы проектирования RESTful API, включая использование HTTP-методов (GET, POST, PUT, DELETE) для выполнения операций с ресурсами.

**3 Описание программы**

**3.1 Общие сведения**

Данный подраздел посвящен представлению общих сведений о разработанном Web API приложении, предназначенном для управления личными финансами. Здесь будет представлена информация об идентификации программного продукта, используемых технологиях и требованиях к программному окружению для его корректного функционирования.

Разработанное Web API приложение имеет условное обозначение “SmartWallet”. Полное наименование программы: “Web API для мониторинга и управления личными финансами”. Данное наименование отражает основную функциональность и предназначение программного продукта.

Для корректной работы Web API “SmartWallet” требуется наличие следующего программного обеспечения:

* Операционная система: Web API может быть развернут на любой операционной системе, поддерживающей платформу .NET Core. Рекомендуется использование серверных операционных систем, таких как Windows Server или Linux (например, Ubuntu, CentOS), для обеспечения стабильной и надежной работы.
* .NET Runtime: Для запуска Web API требуется установленная среда выполнения .NET Core Runtime версии 8.0.0. Данная среда обеспечивает выполнение скомпилированного кода Web API.
* Веб-сервер: Web API требует наличия веб-сервера для обработки HTTP-запросов. Рекомендуется использование Kestrel в качестве веб-сервера. Kestrel - это высокопроизводительный веб-сервер, который может быть использован как самостоятельно, так и в связке с обратным прокси-сервером, таким как Nginx или Apache.
* СУБД (Система управления базами данных): Для хранения данных о пользователях, транзакциях, категориях и других сущностях используется реляционная СУБД. В качестве СУБД используется PostgreSQL. Необходима установленная и настроенная СУБД с соответствующими правами доступа для Web API.

Web API “SmartWallet” разработан с использованием следующих языков программирования и технологий:

* C#: Основной язык программирования, используемый для написания логики Web API, контроллеров, моделей данных и других компонентов. C# является объектно-ориентированным языком программирования, разработанным Microsoft, и широко используется для создания веб-приложений на платформе .NET.
* .NET Core: Web API разработан на базе .NET Core, платформы для разработки приложений от Microsoft. .NET Core предоставляет широкий набор библиотек и инструментов для создания масштабируемых и надежных веб-приложений.

**3.2 Функциональное назначение**

Основное назначение Web API “SmartWallet” заключается в предоставлении централизованного и унифицированного интерфейса для доступа к данным о финансовых операциях пользователей, что позволяет разработчикам создавать удобные и функциональные приложения для личного финансового планирования и анализа.

Более конкретно, Web API решает следующие классы задач:

* Учет доходов и расходов: Web API предоставляет функциональность для добавления, редактирования и удаления информации о доходах и расходах пользователей. Каждая транзакция может быть связана с определенной категорией (например, “продукты”, “транспорт”, “развлечения”), что позволяет пользователям отслеживать структуру своих расходов.
* Анализ финансовых данных: Web API предоставляет функциональность для анализа данных о расходах пользователей. Это позволяет пользователям выявлять тренды, оптимизировать свои расходы и принимать более обоснованные финансовые решения.
* Управление денежными хранилищами: Web API предоставляет функциональность для управления информацией о денежных хранилищах пользователей (например, “наличные”, “банковский счет”, “кредитная карта”). Пользователи могут добавлять, редактировать и удалять информацию о своих хранилищах, а также отслеживать остаток средств в каждом хранилище.
* Авторизация и аутентификация пользователей: Web API обеспечивает безопасный доступ к данным пользователей посредством реализации механизмов авторизации и аутентификации. Это гарантирует, что только авторизованные пользователи имеют доступ к своим финансовым данным.

Несмотря на широкий спектр решаемых задач, Web API “SmartWallet” имеет определенные функциональные ограничения на применение:

* Отсутствие интеграции с банковскими сервисами: Web API не предоставляет прямой интеграции с банковскими сервисами для автоматического получения информации о транзакциях. Пользователи должны вводить информацию о своих доходах и расходах вручную или импортировать данные из других источников.
* Ограниченные возможности анализа данных: Web API предоставляет базовые возможности анализа данных, такие как просмотр категоризованных трат за месяц. Однако, он не предоставляет продвинутых функций анализа данных, таких как прогнозирование финансовых показателей или выявление аномальных транзакций.
* Отсутствие поддержки мультивалютности: Web API не поддерживает работу с несколькими валютами. Все финансовые операции должны быть представлены в одной валюте.
* Зависимость от стабильного интернет-соединения: Для работы Web API требуется стабильное интернет-соединение. В случае отсутствия интернет-соединения, пользователи не смогут получить доступ к своим финансовым данным или выполнить какие-либо операции.
* Ограничения по масштабируемости: Web API может иметь ограничения по масштабируемости при большом количестве пользователей или объеме данных. В этом случае может потребоваться дополнительная оптимизация и масштабирование инфраструктуры.

**3.3 Описание логической структуры**

*< Должны быть указаны:*

*- алгоритм программы;*

*- используемые методы;*

*- схема бд;*

*- структура программы с описанием функций составных частей и связи между ними;*

*- связи программы с другими программами.*

*Описание логической структуры программы выполняют с учетом текста программы на исходном языке >*

Ниже представлена схема базы данных приложения.

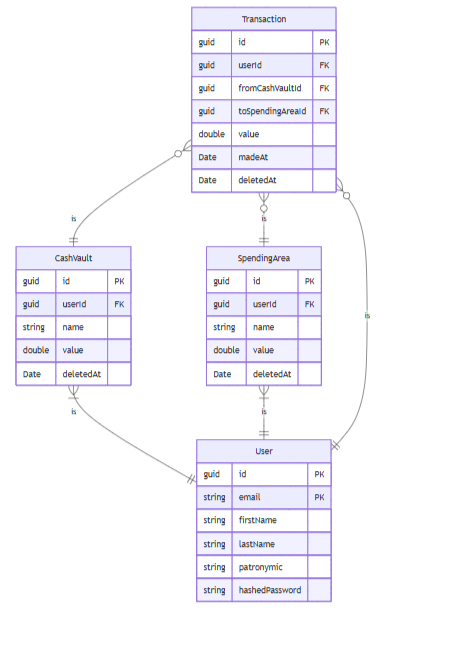


Рисунок 1 - Диаграмма базы данных

**3.4 Используемые технические средства**

*< Должны быть указаны типы электронных вычислительных машин и устройств, которые используются при работе программы.* ***>***

**3.5 Вызов и загрузка**

*< Должны быть указаны: способ вызова программы с соответствующего носителя данных; входные точки в программу. Допускается указывать адреса загрузки, сведения об использовании оперативной памяти, объем программы. >*

**3.6 Входные и выходные данные**

*< Должны быть указаны*: *характер, организация и предварительная подготовка входных данных; формат, описание и способ кодирования входных данных.* ***>***

*< Допускается содержание разделов иллюстрировать пояснительными примерами, таблицами, схемами, графиками. В приложение к описанию программы допускается включать различные материалы, которые нецелесообразно включать в разделы описания. 3.4-3.6*. ***>***

**4 Программа и методика испытаний**

**4.1 Объекты испытаний**

*<Указывают наименование, область применения и обозначение испытуемой программы>*

**4.2 Цель испытаний**

## *<Должна быть указана цель проведения испытаний>*

**4.3 Требования к программе**

<*Должны быть указаны требования, подлежащие проверки во время испытаний и заданные в техническом задании на программу* >

**4.4 Методы испытаний**

<*Должны быть приведены описания используемых методов испытаний. Методы испытаний рекомендуется по отдельным показателям располагать в последовательности, в которой эти показатели расположены в разделе «Требования к программе». В методах испытаний должны быть приведены описания проверок с указанием результатов проведения испытаний (перечней текстовых примеров, контрольных распечаток текстовых примеров и т.п.)*>

**4.5 Тестовый пример**

*<пример оформления тест-кейсов>*

Таблица 1 – Тест – кейс 01 «Тест открытия разработанного раздела»

|  |  |  |  |
| --- | --- | --- | --- |
| Номер тест-кейса | 01 | | |
| Название: | Тест открытия разработанного раздела | | |
| Действие | | Ожидаемый результат | Результат теста:  пройден |
| Предусловие: | | | |
| Запустить сервер с БД, открыть приложение | | Раздел открывается и работает | Пройден |
| Шаги теста: | | | |
| На главной странице ввести логин и пароль | | Авторизация в приложении | Пройден |
| Выбрать из меню слева раздел «Отчеты» | | Выводится информация о работах, доступных для авторизированного пользователя | Пройден |
| Постусловие: | | | |
| Закрыть приложение, выйти из OS | | Приложение закрыто | Пройден |

Таблица 2 – Тест – кейс 02 «Тест адаптивности интерфейса»

|  |  |  |  |
| --- | --- | --- | --- |
| Номер тест-кейса | 02 | | |
| Название: | Тест адаптивности интерфейса | | |
| Действие | | Ожидаемый результат | Результат теста:  пройден |
| Предусловие: | | | |
| Запустить сервер с БД, открыть приложение | | Приложение запущено без ошибок | Пройден |
| Шаги теста: | | | |
| На главной странице ввести логин и пароль | | Авторизация в системе | Пройден |
| Нажать на раздел «Отчеты» | | Раздел открылся без ошибок | Пройден |
| Открыть панель разработчика в браузере (F12) и двигать ползунок ширины экрана до размера мобильного устройства | | Интерфейс адаптировался под мобильное устройство | Пройден |
| Постусловие: | | | |
| Закрыть приложение, выйти из OS | | Приложение закрыто | Пройден |

**5 Руководство оператора**

**5.1 Выполнение программы**

<*Должны быть: указана последовательность действий оператора, обеспечивающих загрузку, запуск, выполнение и завершение программы, приведены описания функций, формата и возможных вариантов команд, с помощью которых оператор осуществляет загрузку и управляет выполнением программы, а также ответы программы на эти команды*>

!!!Обязательны скриншоты окон рабочей программы!!!

**5.2 Сообщение оператору**

*<Должны быть приведены тексты сообщений, выдаваемых в ходе выполнения программы, описание их содержания и соответствующие действия оператора (действия оператора в случае сбоя, возможности повторного запуска программы и т.п.)>*

**Заключение**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Приложение А**

## Текст программы

*<В приложениях к документу могут быть включены таблицы, обоснования, методики, расчеты и другие документы, использованные при разработке>*

**Источники, использованные при разработке**

*<Указывают перечень научно-технических публикаций, нормативно-технических документов и других научно-технических материалов, на которые есть ссылки в основном текст >*

1. ГОСТ Р ИСО\_МЭК 25051-2017 Требования к качеству готового к использованию программного продукта (RUSP) и инструкции по тестированию.
2. ЕСПД Единая система программной документации.
3. Изучаем PHP 7. Руководство по созданию интерактивных веб-сайтов, Дэвид Скляр, 2017
4. Создаем динамические веб-сайты с помощью PHP, MySQL, JavaScript, CSS и HTML5, Никсон Робин, 2019
5. Изучаем JavaScript. Руководство по созданию современных веб-сайтов, Этан Браун, 2017
6. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.
7. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Internet – ресурсы**

1. PHP: Hypertext preprocessor [Электронный ресурс] – [http://php.net](http://php.net/)
2. JavaScript Tutorial [Электронный ресурс] – <http://w3schools.com/js>
3. MySQL [Электронный ресурс] - <https://www.mysql.com/>
4. htmlbook.ru | Для тех кто делает сайты [Электронный ресурс] -http://htmlbook.ru/