Министерство образования Московской области

Государственное бюджетное профессиональное образовательное учреждение Московской области «Физико-технический колледж»

ЗАЩИЩЕНО

ОЦЕНКА\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Руководитель специальности 09.02.07 Информационные

системы и программирование

\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Курсовой проект

Тема: «Разработка системы анализа доходности кроптоактивов, управления криптовалютными портфелями»

**по ПМ.05 Разработка и администрирование систем искусственного интеллекта**

Руководитель КП Разработал

преподаватель студент группы ИСП 4-5

ГБПОУ МО «Физтех-колледж»

\_\_\_\_\_\_\_\_\_\_ Г.В. Базяк \_\_\_\_\_\_\_\_\_\_А.В. Кириллов

Долгопрудный, 2024

СОДЕРЖАНИЕ

[Введение 3](#_Toc184646326)

[1. Роль криптовалютного рынка в современной экономике 3](#_Toc184646327)

[2. Задачи и вызовы управления криптовалютным портфелем 4](#_Toc184646328)

[3. Значение анализа доходности криптоактивов 5](#_Toc184646329)

[4. Основные функции системы анализа и управления портфелем криптовалют 6](#_Toc184646330)

[1 Теоретическая часть 8](#_Toc184646331)

[1.1 Анализ предметной области 8](#_Toc184646332)

[1.2 Определение требований к системе 13](#_Toc184646333)

[2 Проектная часть 14](#_Toc184646334)

[2.1 Сбор и подготовка данных 14](#_Toc184646335)

[2.2 Разработка архитектуры проекта на Django 15](#_Toc184646336)

[2.3 Разработка и обучение модели машинного обучения 15](#_Toc184646337)

[Выбор модели: 15](#_Toc184646338)

[2.4 Интеграция модели в проект Django 16](#_Toc184646339)

[Заключение 16](#_Toc184646340)

[Список используемых источников 19](#_Toc184646341)

[Приложение А 21](#_Toc184646342)

# 

# Введение

Рынок криптовалют в последние годы стал одним из самых динамично развивающихся сегментов финансового мира. С момента появления первой криптовалюты — биткойна — в 2009 году, количество цифровых активов значительно выросло, и сегодня рынок насчитывает тысячи различных криптовалют, обладающих уникальными характеристиками и различными вариантами применения. Основными особенностями криптовалютного рынка являются его высокая волатильность, быстрая изменчивость и отсутствие централизованного регулирования, что делает его крайне привлекательным для инвесторов, ищущих возможности для высокой доходности, но также и сопряженным с высокими рисками. В связи с этим управление криптовалютными портфелями и анализ доходности активов стали ключевыми аспектами успешного инвестирования в этот класс активов.

1. Роль криптовалютного рынка в современной экономике

Криптовалюты изменили традиционные подходы к экономике и финансам, став одним из первых массовых примеров применения технологии блокчейн. Блокчейн, лежащий в основе криптовалют, обеспечивает высокую степень безопасности и децентрализованность, что способствует развитию децентрализованных финансовых систем (DeFi) и экономических моделей, не зависящих от центральных банков и правительств. Привлекательность криптовалютного рынка для профессиональных инвесторов и частных пользователей заключается в возможности получить высокий доход, и многие видят в этом способ защитить свои сбережения от инфляции и нестабильности традиционных финансовых систем.

Сегодня рынок криптовалют предлагает широкий спектр активов с различной степенью доходности, ликвидности и риска. К числу наиболее популярных криптовалютных активов относятся такие криптовалюты, как биткойн, эфириум, а также новые, активно развивающиеся токены и альткоины. Многие из них служат не только средством обмена, но и выполняют роль "смарт-контрактов" и "стейблкоинов," что привлекает все большее внимание со стороны компаний и частных инвесторов. Однако, несмотря на все эти возможности, высокая волатильность криптовалютного рынка создает значительные трудности в управлении инвестициями. Для того чтобы успешно ориентироваться в изменчивой среде и минимизировать риски, инвесторам необходимы инструменты анализа доходности и управления своими криптовалютными активами.

2. Задачи и вызовы управления криптовалютным портфелем

Управление портфелем криптовалют требует глубокого понимания рынка, регулярного анализа динамики цен и учета волатильности активов. Одной из особенностей криптовалютного рынка является его круглосуточная работа, что добавляет сложности в оперативном реагировании на изменения. В условиях постоянных колебаний цен инвесторы вынуждены оперативно перераспределять активы, чтобы минимизировать убытки и увеличить потенциальную прибыль. Для успешного управления портфелем необходимы не только навыки и знания, но и доступ к качественным аналитическим данным о состоянии и изменении рынка.

Наиболее эффективные стратегии управления криптовалютными портфелями обычно включают в себя:

* Анализ исторических данных о стоимости и объемах торгов криптоактивов;
* Построение прогноза на основе технического и фундаментального анализа;
* Балансировку активов для снижения рисков;
* Мониторинг рыночных новостей, влияющих на стоимость активов.

Многие частные инвесторы, особенно новички, часто сталкиваются с трудностями в выборе подходящих криптоактивов и построении сбалансированного портфеля. Для поддержки принятия решений инвесторам необходим доступ к инструментам для анализа данных, которые позволяют быстро и объективно оценить доходность криптовалютного портфеля.

Анализ доходности криптоактивов является неотъемлемой частью успешного инвестирования. В отличие от традиционных акций и облигаций, криптовалюты характеризуются более резкими изменениями стоимости в короткие периоды, что требует использования специализированных методов анализа. Для точной оценки доходности криптовалют необходимо учитывать такие факторы, как средняя цена покупки, текущая рыночная цена, объем торгов, уровни поддержки и сопротивления, а также влияние внешних событий, таких как новости, изменения законодательства и развитие инфраструктуры криптовалютных сетей.

На практике анализ доходности криптоактивов позволяет инвесторам:

* Понимать текущую ценность и перспективы активов в их портфеле;
* Прогнозировать возможные изменения стоимости криптовалют на основе исторических данных;
* Разрабатывать эффективные стратегии для входа и выхода из позиций;
* Оценивать доходность портфеля и его изменения в зависимости от времени и рыночных условий.

Для анализа доходности криптоактивов используются разнообразные метрики и методы. Среди наиболее распространенных методов можно выделить анализ на основе показателей ROI (Return on Investment), годовой доходности, альфа и бета-коэффициентов, а также коэффициентов Шарпа и Сортиного, которые позволяют оценить соотношение доходности и риска. Разработка и использование системы для анализа доходности криптовалютных активов способствуют более осознанному инвестированию, поскольку предоставляют пользователям ключевые данные для оценки их портфеля.

## 3. Основные функции системы анализа и управления портфелем криптовалют

В рамках данной курсовой работы было разработано веб-приложение, которое предоставляет пользователям возможность вводить данные о своих криптоактивах, анализировать их рыночную стоимость и оценивать доходность портфеля за определенные временные периоды. Одной из ключевых особенностей приложения является возможность учитывать объемы торгов, что позволяет более точно отслеживать ликвидность и изменчивость активов, а также понимать, какие активы более активно торгуются на рынке. В приложении реализованы функции для проведения более детализированного анализа, включающие построение графиков динамики цен, отображение объемов торгов и расчет статистики доходности.

Функциональные возможности системы включают:

* Ввод данных о приобретенных криптоактивах (цена покупки, количество, дата);
* Отображение актуальной рыночной стоимости и объемов торгов по каждому активу;
* Анализ доходности портфеля за выбранный период времени;
* Построение графиков, показывающих динамику изменения цен и объемов торгов криптовалют;
* Формирование отчетов и расчет показателей для оценки соотношения риска и доходности.

Эти функции позволяют пользователям детально анализировать доходность их портфеля и принимать более взвешенные инвестиционные решения. Приложение ориентировано на пользователей с различным уровнем подготовки — от начинающих до опытных инвесторов — и предоставляет интуитивно понятный интерфейс для удобного управления данными.

Цель работы

Разработать функциональное и удобное веб-приложение для эффективного управления криптоактивами, отслеживания их стоимости в реальном времени и оценки доходности инвестиций за различные периоды.

Задачи

1. Сделать анализ предметной области
2. Определить требования к системе
3. Описать этапы разработки

# 1 Теоретическая часть

# 1.1 Анализ предметной области

Криптоактивы представляют собой относительно новый класс цифровых финансовых активов, основанных на технологии блокчейн. Основным отличием криптовалют от традиционных финансовых инструментов является их децентрализованный характер, то есть отсутствие центрального регулятора или эмитента, который бы контролировал выпуск и обращение этих активов. В отличие от традиционных денег, которые печатаются и управляются государственными органами и центральными банками, криптовалюты функционируют на основе алгоритмов и консенсусных механизмов, таких как Proof of Work (PoW) или Proof of Stake (PoS), что обеспечивает безопасность и неизменяемость транзакций.

Первая и самая известная криптовалюта — Bitcoin (BTC) — была создана в 2009 году таинственной личностью или группой лиц под псевдонимом Сатоши Накамото. С тех пор рынок криптовалют значительно вырос, и теперь включает тысячи различных активов, известных как альткоины (от «альтернативные коин»). Среди них самые известные и популярные — Ethereum (ETH), Ripple (XRP), Litecoin (LTC), а также новые проекты, такие как Polkadot (DOT), Cardano (ADA) и Solana (SOL). Каждый из этих криптоактивов предлагает уникальные возможности и решения, связанные с различными аспектами блокчейн-технологий, будь то смарт-контракты, децентрализованные приложения или быстрые транзакции.

С момента появления Bitcoin, рынок криптовалют значительно эволюционировал, а вместе с ним и технологии, поддерживающие эти активы. Например, если Bitcoin изначально разрабатывался как альтернатива фиатным валютам и рассматривался как средство сбережения («цифровое золото»), то Ethereum привнес концепцию смарт-контрактов — программируемых алгоритмов, которые могут автоматизировать выполнение договоренностей между участниками сети без участия третьих сторон. Это стало началом для создания децентрализованных финансов (DeFi) и целой экосистемы децентрализованных приложений (dApps), что значительно расширило возможности использования блокчейн-технологий.

Криптоиндустрия продолжает развиваться, и на рынок выходят новые проекты, предлагающие еще более эффективные и масштабируемые решения. Одним из таких перспективных проектов является экосистема TON (The Open Network), которая активно развивается и предлагает уникальный набор функций.

TON (The Open Network) — это децентрализованная блокчейн-платформа, изначально разработанная командой мессенджера Telegram, с целью создания высокопроизводительной и масштабируемой блокчейн-системы. Несмотря на то, что проект столкнулся с регуляторными трудностями, вследствие которых Telegram официально вышел из разработки сети в 2020 году, развитие платформы продолжилось сообществом энтузиастов. TON изначально задумывалась как платформа для массового использования, способная обслуживать миллионы транзакций в секунду и предоставлять децентрализованные услуги широкой аудитории.

Основные характеристики TON:

TON изначально спроектирован для поддержания высоких нагрузок благодаря возможности горизонтального масштабирования. Это достигается через использование уникальной архитектуры, где блокчейн разделяется на несколько подцепочек (shards), каждая из которых может обрабатывать транзакции параллельно, что повышает общую пропускную способность сети.

Благодаря использованию технологии sharding и механизма Proof of Stake (PoS), сеть TON обеспечивает быструю обработку транзакций. Это делает её привлекательной для пользователей, которые нуждаются в скоростных микроплатежах или обработке большого количества операций в короткие сроки.

Одним из ключевых преимуществ TON является его потенциальная интеграция с экосистемой Telegram, что открывает возможности для широкого применения блокчейна среди пользователей этого популярного мессенджера. В перспективе это может привести к массовому внедрению криптовалют и блокчейн-приложений в повседневную жизнь.

TON поддерживает разработку смарт-контрактов и децентрализованных приложений (dApps), что позволяет создавать различные сервисы на базе этой платформы — от финансовых приложений до социальных сетей и игр.

TON предоставляет инструменты для создания децентрализованных приложений (dApps), включая языки программирования для разработки смарт-контрактов, что делает её привлекательной для разработчиков, стремящихся создать масштабируемые и безопасные решения на базе блокчейна.

В рамках экосистемы TON также разрабатываются такие сервисы, как TON DNS — система доменных имен, которая позволяет пользователям легко находить сервисы и контракты в блокчейне через привычные имена. TON Storage предоставляет возможность децентрализованного хранения данных, что делает сеть еще более привлекательной для создания различных приложений и сервисов.

Сеть TON поддерживает создание и использование криптовалютных токенов. Основной токен сети — TON Crystal (или сокращенно TON), который используется для оплаты транзакций, работы смарт-контрактов и вознаграждений валидаторам. Это позволяет TON стать полноценной экосистемой для финансовых операций, платежей и обмена ценностями в децентрализованной среде.

TON позиционируется как платформа для создания многофункциональных децентрализованных сервисов. Помимо базовых функций блокчейна, таких как быстрые транзакции и поддержка смарт-контрактов, экосистема TON включает несколько ключевых компонентов:

Платформа для децентрализованных сервисов, где разработчики могут создавать и предлагать свои приложения, будь то финансовые инструменты, игровые платформы или социальные сети.

Это система микроплатежей, которая позволяет проводить мгновенные и безопасные транзакции между пользователями без необходимости доверять третьим сторонам. TON Payments разработан с целью интеграции в такие платформы, как Telegram, что может значительно упростить использование криптовалют для широкого круга пользователей.

Децентрализованная система доменных имен, которая позволяет пользователям легко взаимодействовать с децентрализованными сервисами и смарт-контрактами через удобные для чтения доменные имена, подобные тем, что используются в традиционном интернете.

Это решение для децентрализованного хранения данных, которое предоставляет возможность хранить большие объемы информации в сети TON, гарантируя безопасность и отказоустойчивость данных.

Децентрализованная система прокси, которая позволяет пользователям получать доступ к различным сервисам в блокчейне анонимно и безопасно, обходя цензуру и другие ограничения.

TON остается одним из самых перспективных блокчейн-проектов на рынке благодаря своей технологической архитектуре и потенциалу для интеграции с такими крупными платформами, как Telegram. Система может оказать значительное влияние на массовое принятие блокчейна и криптовалют, особенно если удастся реализовать задуманные функции по интеграции с мессенджером Telegram.

На текущий момент экосистема TON активно развивается сообществом, что подтверждается ростом числа приложений и сервисов на базе этой платформы. В ближайшем будущем можно ожидать дальнейшего расширения функционала TON, особенно в области децентрализованных финансов (DeFi), приложений для хранения данных и микроплатежей.

В заключение, TON и его экосистема предлагают уникальный набор возможностей для пользователей и разработчиков, позволяя создать высокопроизводительные децентрализованные приложения, проводить мгновенные транзакции и безопасно хранить данные. Это делает TON значимым игроком на рынке криптовалют и блокчейн-технологий.

Криптовалюты и криптоактивы представляют собой децентрализованные цифровые активы, которые работают на основе блокчейн-технологий. Их стоимость может резко меняться, что требует постоянного мониторинга. Особенности, которые следует учитывать:

• Волатильность криптовалютного рынка.

• Отсутствие центрального органа управления.

• Высокий риск, связанный с криптоинвестициями.

• Разнообразие активов: от Bitcoin до новых альткоинов.

Проблемы, с которыми сталкиваются криптоинвесторы

Инвесторы на криптовалютном рынке сталкиваются с рядом сложностей:

• Отслеживание большого количества активов.

• Частое обновление данных из-за волатильности.

• Анализ распределения активов в портфеле.

• Расчет доходности и прогнозирование будущих изменений.

Необходимость автоматизации анализа криптоактивов

Традиционные способы отслеживания криптоактивов (через биржи, таблицы) неэффективны. Требуется автоматизированное решение, которое позволяет инвесторам получать актуальные данные, строить графики доходности и распределения, а также упрощает принятие решений.

# 1.2 Определение требований к системе

Для достижения поставленной цели необходимо решить следующие задачи:

Проектирование и разработка интуитивно понятного интерфейса, через который пользователи смогут вводить данные о своих криптоактивах, получать информацию о состоянии портфеля и отслеживать изменения в стоимости активов. Подключение к различным API для получения актуальных данных о стоимости криптовалют. Это позволит пользователям видеть реальную стоимость активов в режиме реального времени.

Внедрение функционала визуализации данных, который будет показывать изменение стоимости активов за выбранный временной период в виде графиков.

Разработка логики расчета доходности по каждому криптоактиву и по всему портфелю в целом, с учетом изменений стоимости за определенный период.

Добавление возможности учитывать операции покупки и продажи криптовалют, а также ввод и вывод средств, для точного расчета общей доходности портфеля.

Предоставление пользователю возможности визуально оценить распределение активов в портфеле (например, по доле каждой валюты) и предложить рекомендации по возможной оптимизации.

Добавление возможности анализа исторических данных для прогноза будущих изменений стоимости активов на основе выбранных временных периодов и тенденций.

Реализация механизмов безопасности для защиты персональных данных пользователей и информации о их криптоактивах.

Разработка веб-приложения требует использования современных технологий для обеспечения высокой производительности, безопасности и удобства использования. Основными техническими аспектами проекта являются:

Для создания интуитивного пользовательского интерфейса планируется использовать HTML, CSS и JavaScript. Серверная часть приложения будет разрабатываться с использованием Python и фреймворка Django. Серверная логика будет отвечать за обработку данных, расчеты доходности и взаимодействие с базами данных. Для получения актуальных данных о стоимости криптовалют будут использоваться API сторонних сервисов, таких как CoinGecko. Эти источники предоставляют информацию о котировках криптоактивов в реальном времени.

# 2 Проектная часть

## 2.1 Сбор и подготовка данных

Для сбора данных использовались API популярных криптовалютных бирж, таких как CoinGecko. Данные собирались с интервалом в 1 час, что позволяет анализировать краткосрочные и среднесрочные тенденции на рынке криптовалют. В качестве основного источника данных был выбран CoinGecko API, который предоставляет обширную информацию о ценах, объемах торгов и рыночной капитализации криптовалют. Данные собирались с интервалом в 1 час, что позволяет отслеживать динамику рынка в режиме реального времени. Данные были приведены к единому формату, удобному для дальнейшего анализа и обучения моделей. Это включало нормализацию данных, удаление пропусков и приведение всех значений к числовому формату. Учитывались ограничения бесплатной версии CoinGecko API, которые предоставляют данные только за последний год. Для более длительного анализа данных, возможно, потребуется использование платных планов или дополнительных источников данных.

## 2.2 Разработка архитектуры проекта на Django

Были созданы модели Django для хранения данных о криптоактивах, включая цены, объемы торгов и рыночную капитализацию. Каждая модель была настроена для эффективного хранения и извлечения данных. Разработан веб-интерфейс на основе Django, который позволяет пользователям вводить свои криптоактивы, просматривать полную стоимость активов, графики изменения цены портфеля, график Bitcoin тренда, график распределения активов и график объема торгов. Для визуализации данных использовались библиотеки, такие как Matplotlib и Plotly, которые позволяют создавать интерактивные графики и диаграммы. Пользователи могут вводить свои криптоактивы, указывая количество и тип актива. Веб-интерфейс позволяет пользователям просматривать полную стоимость своих активов, а также динамику изменения цены портфеля. Предоставляются графики изменения цены портфеля, тренда Bitcoin, распределения активов и объема торгов.

## 2.3 Разработка и обучение модели машинного обучения

## Выбор модели:

В качестве модели машинного обучения была выбрана рекуррентная нейронная сеть (RNN) на основе LSTM (Long Short-Term Memory), которая хорошо подходит для анализа временных рядов. LSTM позволяет учитывать долгосрочные зависимости в данных, что важно для прогнозирования цен на криптовалюты. Данные были разделены на обучающую и тестовую выборки в соотношении 80/20. Это позволило оценить качество модели на данных, которые не использовались при обучении. Для улучшения качества обучения использовались методы нормализации и масштабирования данных. Это помогло устранить влияние шума и выбросов в данных. Модель обучалась на исторических данных о ценах на Bitcoin с использованием библиотеки TensorFlow. Обучение проводилось с использованием алгоритма обратного распространения ошибки и оптимизатора Adam. Качество модели оценивалось с помощью метрик MSE (Mean Squared Error) и MAE (Mean Absolute Error) на тестовой выборке. Эти метрики позволяют оценить точность прогнозов модели. После обучения модель была сохранена в формате H5 для дальнейшего использования в проекте. Этот формат позволяет легко загружать и использовать модель в веб-приложении.

## 2.4 Интеграция модели в проект Django

Разработан API для прогнозирования, который принимает на вход текущие данные о цене, объеме торгов и рыночной капитализации Bitcoin и возвращает прогноз цены на следующий час. API был интегрирован в проект Django, что позволяет пользователям получать прогнозы непосредственно через веб-интерфейс. API принимает на вход текущие данные о цене, объеме торгов и рыночной капитализации Bitcoin. Возвращает прогноз цены на следующий час, основанный на обученной модели LSTM. Пользователи могут использовать API через веб-интерфейс, вводя текущие данные и получая прогнозы. Возможно автоматическое использование API для получения прогнозов в режиме реального времени, что позволяет отслеживать динамику рынка и принимать обоснованные решения. В проекте реализована система логирования, которая позволяет отслеживать запросы к API и результаты прогнозов. Для обеспечения безопасности, API защищен аутентификацией, что предотвращает несанкционированный доступ.

# Заключение

Разработка веб-приложения для анализа криптоактивов представляет собой актуальную задачу в условиях стремительного роста популярности цифровых валют и децентрализованных финансовых технологий. С увеличением числа людей, инвестирующих в криптовалюты, возникает необходимость в удобных и функциональных инструментах для управления портфелями, оценки доходности и анализа рыночных трендов. Текущие решения зачастую либо не предоставляют необходимой глубины анализа, либо перегружены сложными интерфейсами, что затрудняет их использование, особенно для начинающих пользователей. Таким образом, создание эффективного и интуитивно понятного инструмента становится важным шагом к удовлетворению потребностей современных инвесторов.

Реализованное в рамках курсовой работы веб-приложение предложит пользователям возможность эффективно управлять своими криптовалютными активами, обеспечивая доступ к актуальным данным о стоимости активов, графикам изменений, а также показателям доходности за определенные периоды времени. Это позволит не только отслеживать текущее состояние портфеля, но и анализировать его динамику, что способствует принятию обоснованных решений в отношении дальнейшего распределения активов.

Одним из ключевых преимуществ разрабатываемого приложения станет его фокус на персонализации аналитики. Это особенно важно для криптовалютного рынка, который отличается высокой волатильностью и требует от пользователей регулярного пересмотра своих инвестиционных стратегий. Благодаря возможности получения индивидуализированных рекомендаций и аналитики пользователи смогут принимать более точные решения о том, когда и как перераспределять свои активы для максимальной прибыли или минимизации рисков.

Кроме того, важной составляющей проекта является удобство интерфейса. Приложение будет разрабатываться с учетом потребностей как опытных пользователей, так и новичков на криптовалютном рынке. Это значит, что интерфейс будет интуитивно понятным, а функциональные возможности — доступными, что существенно снизит порог входа для начинающих инвесторов. При этом опытные пользователи получат возможность гибко настраивать аналитику под свои потребности.

Технологическая основа приложения также сыграет важную роль в его успехе. Использование современных веб-технологий позволит обеспечить высокую производительность и стабильность работы системы даже при больших объемах данных и многозадачности. Безопасность приложения станет приоритетом, учитывая чувствительность данных, связанных с криптовалютными активами. Применение передовых методов защиты данных обеспечит высокую степень надежности для пользователей.

Таким образом, создание данного веб-приложения решит множество актуальных задач в сфере управления криптоактивами. Это будет полезный инструмент для широкого круга пользователей, от тех, кто только начинает свой путь в криптовалютах, до профессиональных инвесторов, которым важна гибкость и точность аналитики. В результате выполнения данной курсовой работы будет создано решение, которое поможет инвесторам не только отслеживать и анализировать текущие активы, но и принимать обоснованные, информированные решения на основе объективных данных, что значительно повысит их шансы на успешное управление своими инвестициями.

# Список используемых источников

1. CoinMarketCup [Электронный ресурс] - URL: <https://coinmarketcap.com/> (дата обращения 03.12.2024)
2. Django [Электронный ресурс] - URL: <https://docs.djangoproject.com/en/5.1/>(дата обращения 03.12.2024)
3. LSTM [Электронный ресурс] - URL: <https://keras.io/api/layers/recurrent_layers/lstm/>(дата обращения 03.12.2024)
4. AJAX [Электронный ресурс] - URL: <https://habr.com/ru/articles/862780/>(дата обращения 03.12.2024)
5. html documentation [Электронный ресурс] - URL: <https://developer.mozilla.org/ru/docs/Web/HTML>(дата обращения 03.12.2024)
6. css documentation [Электронный ресурс] - URL: <https://developer.mozilla.org/ru/docs/Web/CSS>(дата обращения 03.12.2024)
7. Chart.js [Электронный ресурс] - URL: <https://www.chartjs.org/docs/latest/samples/information.html>(дата обращения 03.12.2024)
8. Начинаем работать с Chart.js[Электронный ресурс] - URL: <https://code.tutsplus.com/ru/getting-started-with-chartjs-line-and-bar-charts--cms-28384t>(дата обращения 03.12.2024)
9. Models Django [Электронный ресурс] - URL: <https://docs.djangoproject.com/en/5.1/topics/db/models/>(дата обращения 03.12.2024)
10. Javascript documentation [Электронный ресурс] – URL: https://developer.mozilla.org/ru/docs/Web/JavaScript(дата обращения 03.12.2024)

# Приложение А

{% load static %}  
<!DOCTYPE html>  
<html lang="ru">  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1.0">  
 <title>{% block title %}#{% endblock %}</title>  
 {% block style %}<link rel="stylesheet" href="{% static 'style1.css' %}">{% endblock %}  
 <link href="https://cdn.jsdelivr.net/npm/select2@4.1.0-rc.0/dist/css/select2.min.css" rel="stylesheet" />  
</head>  
  
<body>  
<header>  
 <nav>  
 <ul>  
 <li><a href="{% url 'home' %}">Главная</a></li>  
 {% if request.user.is\_authenticated %}  
 <li><a href="{% url 'profile' %}">Профиль</a></li>  
 <li><a href="{% url 'logout' %}">Выйти</a></li>  
 <li>{{ user.username }}</li>  
 {% else %}  
 <li><a href="{% url 'regis' %}">Регистрация</a></li>  
 <li><a href="{% url 'login' %}">Войти</a></li>  
 {% endif %}  
 </ul>  
 </nav>  
</header>  
{% block content %}  
{% endblock %}  
<footer>  
 <div class="container">  
 <p>&copy; 2024 Crypto Currency Evaluation</p>  
 </div>  
</footer>  
<script>  
<!-- Select2 CSS -->  
<link href="https://cdn.jsdelivr.net/npm/select2@4.1.0-rc.0/dist/css/select2.min.css" rel="stylesheet" />  
<!-- Select2 JS -->  
<script src="https://cdn.jsdelivr.net/npm/select2@4.1.0-rc.0/dist/js/select2.min.js"></script>  
</script>

{% extends 'base.html' %}  
{% load static %}  
{% block title %}  
HOME  
{% endblock %}  
<h1>{{ error }}</h1>  
{% block content %}  
<body>  
 <main class="container">  
 <section class="crypto-list">  
 <h2>Add Crypto Currency</h2>  
 <form method="post">  
 {% csrf\_token %}  
 {{ form.as\_p }}  
 <button type="submit">Add</button>  
 </form>  
 </section>  
 </main>  
</body>  
  
<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>  
<script src="https://cdn.jsdelivr.net/npm/select2@4.1.0-rc.0/dist/js/select2.min.js"></script>  
<link href="https://cdnjs.cloudflare.com/ajax/libs/select2/4.1.0/css/select2.min.css" rel="stylesheet">  
<script>  
 $(document).ready(function() {  
 $('.select2').select2({  
 placeholder: "Search for a coin",  
 minimumInputLength: 2,  
 ajax: {  
 url: "{% url 'search\_coins' %}", // URL для AJAX-запроса  
 dataType: 'json',  
 delay: 250,  
 data: function (params) {  
 return {  
 q: params.term // Текст, введённый пользователем  
 };  
 },  
 processResults: function (data) {  
 return {  
 results: data.results // Форматирование результата  
 };  
 },  
 cache: true  
 }  
 });  
 });  
</script>  
{% endblock %}

{% extends 'base.html' %}  
{% load static %}  
{% block title %}Ваш портфель{% endblock %}  
{% block style %}  
<link rel="stylesheet" href="{% static 'profile\_style.css' %}">  
{% endblock %}  
{% block content %}  
<div class="portfolio-container">  
 {% if error %}  
 <p class="error-message">{{ error }}</p>  
 {% else %}  
 <!-- Заголовок -->  
 <h1 class="portfolio-title">Портфель пользователя {{ portfolio.user.username }}</h1>  
 <!-- Таблица с данными портфеля -->  
 <table class="portfolio-table">  
 <thead>  
 <tr>  
 <th>Монета</th>  
 <th>Количество</th>  
 <th>Цена (USD)</th>  
 <th>Общая стоимость (USD)</th>  
 </tr>  
 </thead>  
 <tbody>  
 {% for coin in portfolio\_coins %}  
 <tr>  
 <td>{{ coin.coin.name }}</td>  
 <td>{{ coin.amount|floatformat:3 }}</td>  
 <td>${{ coin.price|floatformat:3 }}</td>  
 <td>${{ coin.total\_value|floatformat:3 }}</td>  
 </tr>  
 {% endfor %}  
 </tbody>  
 </table>  
 <!-- Форма для управления монетами -->  
 <form method="post" action="{% url 'manage\_coin' %}">  
 {% csrf\_token %}  
 <div class="form-group">  
 <!-- Выпадающий список для выбора монеты -->  
 <label for="coin\_id">Выберите монету:</label>  
 <select id="coin\_id" name="coin\_id" required>  
 {% for coin in portfolio\_coins %}  
 <option value="{{ coin.coin.id }}">{{ coin.coin.name }}</option>  
 {% endfor %}  
 </select>  
 </div>  
  
 <div class="form-group">  
 <!-- Поле для ввода количества -->  
 <label for="amount">Количество:</label>  
 <input type="number" step="any" id="amount" name="amount" placeholder="Введите количество" required>  
 </div>  
 <div class="form-actions">  
 <!-- Кнопки действий -->  
 <button type="submit" name="action" value="update">Обновить</button>  
 <button type="submit" name="action" value="sell">Продать</button>  
 </div>  
 </form>  
 <!-- Отображение общей стоимости портфеля -->  
 <div class="total-value">  
 <h3>Общая стоимость портфеля: ${{ total\_value|floatformat:3 }}</h3>  
 </div>  
 {% endif %}  
</div>  
<!-- Графики -->  
<div class="chart-container">  
 <h2>График изменения стоимости портфеля</h2>  
 <canvas id="portfolioChart"></canvas>  
</div>  
<div class="chart-container">  
 <h2>График тренда Bitcoin</h2>  
 <canvas id="bitcoinChart"></canvas>  
</div>  
<div class="chart-container">  
 <h2>Распределение активов</h2>  
 <canvas id="pieChart"></canvas>  
</div>  
<!-- Подключение Chart.js -->  
<script src="https://cdn.jsdelivr.net/npm/chart.js"></script>  
<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>  
<script>  
 // Функция для обновления графика  
 function updateChart(data, chartId) {  
 var chart = Chart.getChart(chartId);  
 chart.data.labels = data.timestamps;  
 chart.data.datasets[0].data = data.values;  
 chart.update();  
 }  
 // Функция для первоначальной загрузки данных и инициализации графика  
 function initChart(url, chartId, chartType, label) {  
 $.ajax({  
 url: url,  
 method: "GET",  
 success: function(data) {  
 console.log("Initial data:", data);  
 // Инициализация графика с первоначальными данными  
 var ctx = document.getElementById(chartId).getContext('2d');  
 new Chart(ctx, {  
 type: chartType,  
 data: {  
 labels: data.timestamps,  
 datasets: [{  
 label: label,  
 data: data.values,  
 borderColor: 'rgba(75, 192, 192, 1)',  
 tension: 0.1  
 }]  
 }  
 });  
 }  
 });  
 }  
 // Обновление графика каждые 5 секунд  
 setInterval(function() {  
 $.ajax({  
 url: "{% url 'get\_portfolio\_history' %}",  
 method: "GET",  
 success: function(data) {  
 console.log("Updating data:", data);  
 updateChart(data, 'portfolioChart');  
 }  
 });  
 $.ajax({  
 url: "{% url 'get\_bitcoin\_price\_history' %}",  
 method: "GET",  
 success: function(data) {  
 console.log("Updating Bitcoin data:", data);  
 updateChart(data, 'bitcoinChart');  
 }  
 });  
 }, 5000)  
 // Инициализация графиков при загрузке страницы  
 $(document).ready(function() {  
 initChart("{% url 'get\_portfolio\_history' %}", 'portfolioChart', 'line', 'Стоимость портфеля');  
 initChart("{% url 'get\_bitcoin\_price\_history' %}", 'bitcoinChart', 'line', 'Bitcoin Price');  
 // График распределения активов  
 var pieCtx = document.getElementById('pieChart').getContext('2d');  
 var pieData = {{ pie\_data|safe }};  
 var pieChart = new Chart(pieCtx, {  
 type: 'pie',  
 data: {  
 labels: pieData.labels,  
 datasets: [{  
 data: pieData.values,  
 backgroundColor: ['#FF6384', '#36A2EB', '#FFCE56']  
 }]  
 }  
 });  
 });  
</script>  
{% endblock %}

from django.contrib.auth import login, authenticate, logout  
from django.http import JsonResponse, HttpResponseBadRequest  
from django.contrib.auth.decorators import login\_required  
from django.shortcuts import render, redirect, get\_object\_or\_404  
from .forms import CustomUserForm, CustomAuthentication, PortfolioCoinForm  
from .models import CoinList, Portfolio, PortfolioCoin, Coin  
import requests  
from django.core.cache import cache  
from .Api\_Coin\_Gekko import coin\_price  
from decimal import Decimal  
from .models import PortfolioHistory  
import json  
from django.core.serializers.json import DjangoJSONEncoder  
from django.views.decorators.http import require\_POST  
from django.http import JsonResponse  
from django.contrib import messages  
from django.utils import timezone  
def get\_portfolio\_value(portfolio):  
 base\_url = 'https://api.coingecko.com/api/v3/simple/price'  
 symbols = [coin.coin.symbol for coin in PortfolioCoin.objects.filter(portfolio=portfolio)]  
 query\_params = {  
 'ids': ','.join(symbols),  
 'vs\_currencies': 'usd'  
 }  
 response = requests.get(base\_url, params=query\_params)  
 if response.status\_code == 200:  
 prices = response.json()  
 total\_value = 0  
 detailed\_values = []  
 for coin in PortfolioCoin.objects.filter(portfolio=portfolio):  
 coin\_price = prices.get(coin.coin.symbol, {}).get('usd', 0)  
 coin\_value = coin.amount \* coin\_price  
 total\_value += coin\_value  
 detailed\_values.append({  
 'coin': coin.coin.name,  
 'symbol': coin.coin.symbol,  
 'amount': coin.amount,  
 'price': coin\_price,  
 'value': coin\_value  
 })  
 return total\_value, detailed\_values  
 else:  
 return 0, []  
def home(request):  
 if not request.user.is\_authenticated:  
 return redirect('login')  
 portfolio, created = Portfolio.objects.get\_or\_create(user=request.user)  
 if request.method == 'POST':  
 form = PortfolioCoinForm(request.POST)  
 if form.is\_valid():  
 coins = form.cleaned\_data['coin'] # Получаем список выбранных монет  
 amount = form.cleaned\_data['amount']  
 for coin in coins: # Перебираем монеты и сохраняем их по одной  
 PortfolioCoin.objects.update\_or\_create(  
 portfolio=portfolio,  
 coin=coin,  
 defaults={'amount': amount}  
 )  
 return redirect('profile') # Перенаправление в личный кабинет  
 else:  
 return render(request, 'home.html', {'form': form, 'error': 'Invalid input'})  
 else:  
 form = PortfolioCoinForm()  
 return render(request, 'home.html', {'form': form})  
def update\_portfolio\_history(portfolio, total\_value):  
 PortfolioHistory.objects.create(  
 portfolio=portfolio,  
 total\_value=total\_value,  
 timestamp=timezone.now()  
 )  
@login\_required  
def profile(request):  
 if not request.user.is\_authenticated:  
 return redirect('regis') # Редирект, если пользователь не авторизован  
 portfolio = Portfolio.objects.filter(user=request.user).first()  
 if not portfolio:  
 return HttpResponseBadRequest("Портфель не найден")  
 # Используем правильный способ получения связанных объектов  
 portfolio\_coins = PortfolioCoin.objects.filter(portfolio=portfolio)  
 # Получаем цену для каждой монеты  
 for portfolio\_coin in portfolio\_coins:  
 coin\_id = portfolio\_coin.coin.currency\_id # Уникальный идентификатор монеты  
 try:  
 # Получаем цену с помощью функции coin\_price\_cached  
 price\_data = coin\_price\_cached(coin\_id)  
 if not price\_data: # Если данные пустые, пропускаем  
 print(f"Не удалось получить данные для {coin\_id}. Пропускаем...")  
 continue  
 price = Decimal(price\_data.get(coin\_id, {}).get('usd', 0)) # Преобразуем в Decimal  
 # Обновляем цену в модели PortfolioCoin  
 portfolio\_coin.price = price  
 portfolio\_coin.save()  
 # Добавляем атрибут `total\_value` для удобства  
 portfolio\_coin.total\_value = portfolio\_coin.amount \* portfolio\_coin.price  
 except Exception as e:  
 print(f"Ошибка получения цены для {coin\_id}: {e}")  
 # Расчет общей стоимости портфеля  
 total\_value = sum(  
 portfolio\_coin.total\_value for portfolio\_coin in portfolio\_coins if portfolio\_coin.price  
 )  
 # Обновляем историю портфеля  
 update\_portfolio\_history(portfolio, total\_value)  
  
 # Обработка POST-запроса  
 if request.method == 'POST':  
 coin\_id = request.POST.get('coin\_id')  
 amount = Decimal(request.POST.get('amount'))  
 action = request.POST.get('action')  
 try:  
 portfolio\_coin = portfolio\_coins.get(coin\_\_id=coin\_id)  
 except PortfolioCoin.DoesNotExist:  
 return HttpResponseBadRequest("Монета не найдена в портфеле")  
 if action == 'update':  
 portfolio\_coin.amount = amount  
 portfolio\_coin.save()  
 elif action == 'sell':  
 if portfolio\_coin.amount < amount:  
 return HttpResponseBadRequest("Недостаточно монет для продажи")  
 elif portfolio\_coin.amount == amount:  
 portfolio\_coin.delete()  
 else:  
 portfolio\_coin.amount -= amount  
 portfolio\_coin.save()  
 return redirect('profile')  
 context = {  
 'portfolio': portfolio,  
 'portfolio\_coins': portfolio\_coins,  
 'total\_value': total\_value,  
 }  
 history = PortfolioHistory.objects.filter(portfolio=portfolio).order\_by('timestamp')  
 history\_data = {  
 'timestamps': [entry.timestamp.strftime('%Y-%m-%d %H:%M:%S') for entry in history],  
 'values': [float(entry.total\_value) for entry in history]  
 }  
 context['history\_data'] = json.dumps(history\_data, cls=DjangoJSONEncoder)  
 pie\_data = {  
 'labels': [coin.coin.name for coin in portfolio\_coins],  
 'values': [float(coin.total\_value) for coin in portfolio\_coins if coin.total\_value]  
 }  
 context['pie\_data'] = json.dumps(pie\_data)  
 return render(request, 'profile.html', context)  
@require\_POST  
@login\_required  
def update\_coin(request):  
 coin\_id = request.POST.get('coin\_id')  
 new\_amount = request.POST.get('new\_amount')  
 portfolio = get\_object\_or\_404(Portfolio, user=request.user)  
 try:  
 portfolio\_coin = PortfolioCoin.objects.get(portfolio=portfolio, coin\_\_id=coin\_id)  
 portfolio\_coin.amount = Decimal(new\_amount)  
 portfolio\_coin.save()  
 except PortfolioCoin.DoesNotExist:  
 return JsonResponse({'error': 'Актив не найден'}, status=400)  
 return redirect('profile')  
@require\_POST  
@login\_required  
def sell\_coin(request):  
 coin\_id = request.POST.get('coin\_id')  
 portfolio = get\_object\_or\_404(Portfolio, user=request.user)  
 try:  
 portfolio\_coin = PortfolioCoin.objects.get(portfolio=portfolio, coin\_\_id=coin\_id)  
 portfolio\_coin.delete() # Удаляем актив из портфеля  
 except PortfolioCoin.DoesNotExist:  
 return JsonResponse({'error': 'Актив не найден'}, status=400)  
 return redirect('profile')  
def coin\_price\_cached(currency\_id):  
 cache\_key = f"coin\_price\_{currency\_id}"  
 price\_data = cache.get(cache\_key) # Получаем данные из кэша  
 if price\_data is None: # Если данных нет, запрашиваем через API  
 price\_data = coin\_price(currency\_id) # Функция возвращает полный словарь  
 if price\_data: # Если словарь не пустой  
 cache.set(cache\_key, price\_data, timeout=300) # Кэшируем полный словарь на 5 минут  
 else:  
 print(f"Не удалось получить данные для {currency\_id}, кэширование пропущено.")  
 return {}  
 return price\_data # Возвращаем словарь или пустой объект  
def search\_coins(request):  
 query = request.GET.get('q', '') # Получаем текст из запроса  
 coins = CoinList.objects.filter(name\_\_icontains=query)[:10] # Ограничиваем результаты  
 results = [{'id': coin.id, 'text': coin.name} for coin in coins] # Формат JSON  
 return JsonResponse({'results': results})  
def UserCreation(request):  
 if request.method == 'POST':  
 form = CustomUserForm(request.POST)  
 if form.is\_valid():  
 user\_save = form.save()  
 login(request, user\_save)  
 return redirect('home')  
 else:  
 return render(request, 'UserCreationForm.html', {'form': form})  
 else:  
 form = CustomUserForm()  
 return render(request, 'UserCreationForm.html', {'form': form})  
def Authenticate(request):  
 if request.method == 'POST':  
 form = CustomAuthentication(request.POST, data=request.POST)  
 if form.is\_valid():  
 user = form.cleaned\_data.get('username')  
 password = form.cleaned\_data.get('password')  
 aunt = authenticate(username=user, password=password)  
 login(request, aunt)  
 return redirect('home')  
 else:  
 print('#####################')  
 print('Error')  
 return render(request, 'AuthenticationForm.html', {'form': form})  
 else:  
 form = CustomAuthentication()  
 return render(request, 'AuthenticationForm.html', {'form': form})  
def \_logout(request):  
 logout(request)  
 return redirect('home')  
@login\_required  
def portfolio(request):  
 portfolio = Portfolio.objects.get(user=request.user)  
 coins = portfolio.coins.all()  
 print(coins)  
 return render(request, 'portfolio.html', {'coins': coins})  
def manage\_coin(request):  
 if request.method == 'POST':  
 coin\_id = request.POST.get('coin\_id')  
 amount = Decimal(request.POST.get('amount'))  
 action = request.POST.get('action')  
 portfolio = Portfolio.objects.filter(user=request.user).first()  
 if not portfolio:  
 return HttpResponseBadRequest("Портфель не найден")  
 try:  
 portfolio\_coin = PortfolioCoin.objects.get(portfolio=portfolio, coin\_\_id=coin\_id)  
 except PortfolioCoin.DoesNotExist:  
 return HttpResponseBadRequest("Монета не найдена в портфеле")  
 if action == 'update':  
 portfolio\_coin.amount = amount  
 portfolio\_coin.save()  
 elif action == 'sell':  
 if portfolio\_coin.amount < amount:  
 return HttpResponseBadRequest("Недостаточно монет для продажи")  
 elif portfolio\_coin.amount == amount:  
 portfolio\_coin.delete()  
 else:  
 portfolio\_coin.amount -= amount  
 portfolio\_coin.save()  
 return redirect('profile')  
 return HttpResponseBadRequest("Неверный метод запроса")  
def get\_portfolio\_history(request):  
 if not request.user.is\_authenticated:  
 return JsonResponse({"error": "Пользователь не авторизован"}, status=401)  
 portfolio = Portfolio.objects.filter(user=request.user).first()  
 if not portfolio:  
 return JsonResponse({"error": "Портфель не найден"}, status=404)  
 history = PortfolioHistory.objects.filter(portfolio=portfolio).order\_by('timestamp')  
 history\_data = {  
 'timestamps': [entry.timestamp.strftime('%Y-%m-%d %H:%M:%S') for entry in history],  
 'values': [float(entry.total\_value) for entry in history]  
 }  
 print(history\_data)  
 return JsonResponse(history\_data)  
def get\_bitcoin\_price\_history(request):  
 url = "https://api.coingecko.com/api/v3/coins/bitcoin/market\_chart"  
 params = {  
 'vs\_currency': 'usd',  
 'days': '60', # Получаем данные за последние 60 дней  
 'interval': 'daily'  
 }  
 response = requests.get(url, params=params)  
 data = response.json()  
 prices = data['prices']  
 timestamps = [price[0] for price in prices]  
 values = [price[1] for price in prices]  
 history\_data = {  
 'timestamps': timestamps,  
 'values': values  
 }  
 return JsonResponse(history\_data)

import requests  
from datetime import datetime  
import csv  
import psycopg2  
# """Coin Geeko API"""  
# {'id': 'bitcoin', 'symbol': 'btc', 'name': 'Bitcoin'}  
COINS = ['bitcoin', 'the-open-network']  
def ping():  
 url = "https://api.coingecko.com/api/v3/ping"  
 headers = {  
 "accept": "application/json",  
 "x-cg-api-key": "CG-Yht7zs2aigen8xwtZY9c1xgP",  
 }  
 response = requests.get(url, headers)  
 data = response.json()  
 return data  
def coins\_list():  
 url = "https://api.coingecko.com/api/v3/coins/list"  
 headers = {  
 "accept": "application/json",  
 "x-cg-api-key": "CG-Yht7zs2aigen8xwtZY9c1xgP",  
 }  
 response = requests.get(url, headers)  
 return response.json()  
def supported\_currencies\_list():  
 url = "https://api.coingecko.com/api/v3/simple/supported\_vs\_currencies"  
 headers = {"accept": "application/json"}  
 response = requests.get(url, headers=headers)  
 return response.json()  
def coin\_price(currency\_id):  
 url\_price = f"https://api.coingecko.com/api/v3/simple/price?ids={currency\_id}&vs\_currencies=usd&precision=4"  
 headers = {  
 "accept": "application/json",  
 "x-cg-api-key": "CG-Yht7zs2aigen8xwtZY9c1xgP",  
 }  
 try:  
 response = requests.get(url\_price, headers=headers)  
 response.raise\_for\_status() # Проверяем HTTP-ошибки  
 data = response.json()  
 return data # Возвращаем полный словарь, например: {'bitcoin': {'usd': 97485.3502}}  
 except requests.exceptions.RequestException as e:  
 print(f"Ошибка запроса к API: {e}")  
 return {} # Возвращаем пустой словарь вместо None  
def coin\_historical\_chart\_data(coin\_id, vs\_currency, date\_from, date\_to):  
 url = (f"https://api.coingecko.com/api/v3/coins/{coin\_id}/market\_chart/range?vs\_currency={vs\_currency}"  
 f"&from={date\_from}&to={date\_to}&precision=3") # interval=daily&  
 headers = {  
 "accept": "application/json",  
 "x-cg-api-key": "CG-Yht7zs2aigen8xwtZY9c1xgP"  
 }  
 response = requests.get(url, headers=headers)  
 return response.json()  
def save\_data\_csv():  
 data = coin\_historical\_chart\_data('the-open-network', 'usd', 1724225702, 1729496102)  
 if 'error' not in data:  
 csv\_file = 'currency\_data.csv'  
 with open(csv\_file, mode='w', newline='', encoding='utf-8') as file:  
 writer = csv.writer(file)  
 writer.writerow(['datetaime', 'price', 'market\_caps', 'total\_volumes'])  
 for i in range(len(data['prices'])):  
 date = datetime.utcfromtimestamp(data['prices'][i][0] / 1000.0)  
 formatted\_date = date.strftime('%Y-%m-%d %H:%M:%S')  
 price = data['prices'][i][1]  
 market\_cap = data['market\_caps'][i][1]  
 total\_volume = data['total\_volumes'][i][1]  
 writer.writerow([formatted\_date, price, market\_cap, total\_volume])  
  
 print(f"Данные были сохранены в {csv\_file}")  
 else:  
 print('Номер ошибки -', data['error']['status']['error\_code'], 'Запись об ошибке -',  
 data['error']['status']['error\_message'])  
def portfolio():  
 currency\_id = input("Введите id валют, которую хотите добавить \n").split()  
 currency\_list = []  
 for i in currency\_id:  
 currency\_list.append(coin\_price(i))  
 print(currency\_list)  
############################## PosgreSQL ###############################  
conn\_params = {  
 'dbname': 'postgres\_django',  
 'user': 'postgres',  
 'password': '1234',  
 'host': 'localhost',  
 'port': '5432'  
}  
def connect():  
 try:  
 conn = psycopg2.connect(\*\*conn\_params)  
 print("Подключение успешно установлено")  
 return conn  
 except psycopg2.Error as e:  
 print(f"Ошибка подключения к базе данных: {e}")  
 exit()  
import csv  
from datetime import datetime, timedelta  
def save\_multiple\_months\_data(start\_timestamp, end\_timestamp, coin\_id='the-open-network', currency='usd',  
 csv\_file='currency\_data.csv'):  
 *"""  
 Сохраняет данные по криптовалюте за несколько месяцев в один файл.  
 :param start\_timestamp: Начальная временная метка (в формате UNIX).  
 :param end\_timestamp: Конечная временная метка (в формате UNIX).  
 :param coin\_id: ID криптовалюты на платформе API.  
 :param currency: Валюта для отображения данных.  
 :param csv\_file: Имя CSV-файла для сохранения.  
 """* try:  
 # Открываем файл для записи (добавление данных, если файл существует)  
 with open(csv\_file, mode='w', newline='', encoding='utf-8') as file:  
 writer = csv.writer(file)  
 # Записываем заголовок файла, если он новый  
 writer.writerow(['datetime', 'price', 'market\_caps', 'total\_volumes'])  
 # Устанавливаем начальную точку  
 current\_start = start\_timestamp  
 while current\_start < end\_timestamp:  
 # Устанавливаем конец текущего месяца  
 current\_end = min(current\_start + 30 \* 24 \* 3600, end\_timestamp)  
 # Получаем данные за период  
 data = coin\_historical\_chart\_data(coin\_id, currency, current\_start, current\_end)  
 if 'error' not in data:  
 for i in range(len(data['prices'])):  
 date = datetime.utcfromtimestamp(data['prices'][i][0] / 1000.0)  
 formatted\_date = date.strftime('%Y-%m-%d %H:%M:%S')  
 price = data['prices'][i][1]  
 market\_cap = data['market\_caps'][i][1]  
 total\_volume = data['total\_volumes'][i][1]  
 writer.writerow([formatted\_date, price, market\_cap, total\_volume])  
 print(f"Данные за период {current\_start} - {current\_end} сохранены.")  
 else:  
 print(f"Ошибка получения данных за период {current\_start} - {current\_end}: ",  
 data['error']['status']['error\_message'])  
 # Переходим к следующему месяцу  
 current\_start = current\_end + 1  
 except Exception as e:  
 print(f"Произошла ошибка: {str(e)}")  
# save\_multiple\_months\_data(1669852761, 1701388761)  
import csv  
from datetime import datetime, timedelta  
def save\_bitcoin\_prices():  
 *"""  
 Сохраняет данные по ценам на Bitcoin за период 2020–2024 в CSV файл.  
 """* start\_date = datetime(2024, 8, 1) # Начало периода  
 end\_date = datetime(2024, 12, 1) # Конец периода (на 1 день меньше конца)  
 coin\_id = 'the-open-network' # ID криптовалюты  
 currency = 'usd' # Валюта  
 csv\_file = 'ton\_prices\_2024\_08-12.csv' # Название файла  
 # Преобразование дат в UNIX timestamp  
 start\_timestamp = int(start\_date.timestamp())  
 end\_timestamp = int(end\_date.timestamp())  
 try:  
 # Создаем файл для записи  
 with open(csv\_file, mode='w', newline='', encoding='utf-8') as file:  
 writer = csv.writer(file)  
 # Записываем заголовок файла  
 writer.writerow(['datetime', 'price', 'market\_caps', 'total\_volumes'])  
 # Устанавливаем начальную точку  
 current\_start = start\_timestamp  
 while current\_start < end\_timestamp:  
 # Устанавливаем конец текущего интервала (максимум 30 дней)  
 current\_end = min(current\_start + 30 \* 24 \* 3600, end\_timestamp)  
 # Получаем данные за период  
 data = coin\_historical\_chart\_data(coin\_id, currency, current\_start, current\_end)  
 if 'error' not in data:  
 for i in range(len(data['prices'])):  
 date = datetime.utcfromtimestamp(data['prices'][i][0] / 1000.0)  
 formatted\_date = date.strftime('%Y-%m-%d %H:%M:%S')  
 price = data['prices'][i][1]  
 market\_cap = data['market\_caps'][i][1]  
 total\_volume = data['total\_volumes'][i][1]  
 writer.writerow([formatted\_date, price, market\_cap, total\_volume])  
 print(f"Данные за период {datetime.utcfromtimestamp(current\_start).strftime('%Y-%m-%d')} "  
 f"- {datetime.utcfromtimestamp(current\_end).strftime('%Y-%m-%d')} сохранены.")  
 else:  
 print(  
 f"Ошибка получения данных за период {datetime.utcfromtimestamp(current\_start).strftime('%Y-%m-%d')} "  
 f"- {datetime.utcfromtimestamp(current\_end).strftime('%Y-%m-%d')}: ",  
 data['error']['status']['error\_message'])  
 # Переходим к следующему интервалу  
 current\_start = current\_end + 1  
 except Exception as e:  
 print(f"Произошла ошибка: {str(e)}")  
# Запуск функции  
# save\_bitcoin\_prices()  
# print(coin\_price('bitcoin'))