МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КубГУ»)**

**Факультет компьютерных технологий и прикладной математики**

**Кафедра вычислительных технологий**

**КУРСОВАЯ РАБОТА**

**МОДЕЛИРОВАНИЕ И ИССЛЕДОВАНИЕ ПОВЕДЕНИЯ АГЕНТОВ С ПОМОЩЬЮ НЕЙРОЭВОЛЮЦИИ**

Работу выполнил \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ М.С. Прозоров

(подпись)

Направление подготовки 02.03.02 — «Фундаментальная информатика и\_\_\_\_\_

(код, наименование)

информационные технологии»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_курс\_\_\_\_\_\_\_\_3\_\_\_\_\_\_\_\_

Направленность (профиль) \_\_\_\_\_\_\_\_\_\_\_\_Фундаментальная информатика и информационные технологии\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Научный руководитель

канд. техн. наук, доц. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Т.А. Приходько

(подпись, дата)

Нормоконтролер

ассистент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Е.А. Нигодин

(подпись, дата)

Краснодар

2022

**РЕФЕРАТ**

Курсовая работа 34 стр., 3 ч., 20 рис., 5 источников, 1 приложение.

ИСКУССТВЕННАЯ ЖИЗНЬ, НЕЙРОННАЯ СЕТЬ, СИМУЛЯТОР, DART, FLUTTER

Цель работы курсовой работы – создать модель искусственной жизни и разработать кроссплатформенный симулятор для этой модели.

Объектом исследования в работе является искусственная жизнь.

Предметом исследования является разработка симулятора искусственной жизни.

В качестве методов исследования использовались эмпирический метод, сбор статистической информации, графический метод и наблюдение. В результате работы были изучены принципы моделирования искусственной жизни.

Научная новизна работы заключается в формировании новой модели искусственной жизни и разработке симулятора для изучения это модели.

По результатам исследования, был разработан кроссплатформенный симулятор с визуализацией и пользовательским интерфейсом, предназначенный для моделирования и исследования поведения агентов.

**СОДЕРЖАНИЕ**

[Введение 4](#_Toc104396660)

[1 Теоретическая информация о искусственной жизне 6](#_Toc104396661)

[1.1 Общие сведения 6](#_Toc104396662)

[1.2 Модели искусственной жизни 7](#_Toc104396663)

[1.3 Адаптивное поведение 9](#_Toc104396664)

[1.4 Структура блока 10](#_Toc104396665)

[1.5 Транзакции 10](#_Toc104396666)

[1.6 Электронная цифровая подпись 12](#_Toc104396667)

[1.7 Майнинг 13](#_Toc104396668)

[1.8 Консенсус 14](#_Toc104396669)

[2 Сферы применения блокчейн сетей 16](#_Toc104396670)

[2.1 Образование 16](#_Toc104396671)

[2.2 Голосование 17](#_Toc104396672)

[2.3 Авторство и право владения 17](#_Toc104396673)

[2.4 Смарт-контракты 17](#_Toc104396674)

[2.5 Электронные медицинские карты 18](#_Toc104396675)

[2.6 Медицинские исследования 18](#_Toc104396676)

[2.7 Цепочки поставок и борьба с контрафактом 19](#_Toc104396677)

[2.8 Благотворительность 19](#_Toc104396678)

[3 Реализация тренажёра 20](#_Toc104396679)

[3.1 Инструменты для разработки 20](#_Toc104396680)

[3.2 Требования к реализации 21](#_Toc104396681)

[3.3 Описание функциональности тренажёра 21](#_Toc104396682)

[Заключение 28](#_Toc104396683)

[Список использованных источников 29](#_Toc104396684)

[Приложение А 30](#_Toc104396685)

# ВВЕДЕНИЕ

Искусственная жизнь — это понятие, введенное Крисом Лангтоном для обозначения множества компьютерных механизмов, которые используются для моделирования естественных систем. Искусственная жизнь применяется для моделирования процессов в экономике, поведения животных и насекомых, а также взаимодействия различных объектов. В настоящий момент искусственная жизнь — это наука, занимающаяся изучением жизни, живых систем и их эволюции при помощи созданных человеком моделей.

Искусственная жизнь может быть описана как теория и практика моделирования биологических систем. Разработчики, которые ведут исследования в данной сфере, надеются, что путем моделирования биологических систем мы сможем лучше понять, почему и как они работают. С помощью моделей разработчики могут управлять созданной средой, проверять различные гипотезы и наблюдать, как системы и среда реагируют на изменения.

В данной курсовой работе рассматривается одно из направлений искусственной жизни — синтетическую науку о поведении. Её очень чётко описывает Брюс МакЛеннан:

«Синтетическая наука о поведении — это подход к изучению поведения животных, при котором простые синтетические организмы определенным образом действуют в синтетическом мире. Так как и мир, и организмы являются синтетическими, они могут быть сконструированы для особых целей, а именно для проверки определенных гипотез»

Помимо исследовательской работы, запланировано разработать симулятор, предназначенный для моделирования и исследования поведения агентов.

Основная цель работы – создать модель искусственной жизни и разработать симулятор для этой модели.

Для реализации поставленной цели предполагается решить следующие задачи:

* изучить принципы моделирования искусственной жизни;
* создать собственную модель искусственной жизни;
* выбрать инструментарий для разработки симулятора и разработать его;

Объектом исследования в работе является искусственная жизнь.

Предметом исследования является разработка симулятора искусственной жизни.

Информационная база исследования включает в себя несколько видов учебных материалов таких как статьи от авторитетных автором, доклады с конференций и различные видеоматериалы с демонстрацией других моделей ИЖ. В качестве методов исследования использовались абстрагирование, наблюдение, моделирование и эксперименты.

Научная новизна работы заключается в формировании новой модели искусственной жизни и разработке симулятора для изучения это модели.

По результатам исследования, был разработан кроссплатформенный симулятор с визуализацией и пользовательским интерфейсом, предназначенный для моделирования и исследования поведения агентов.

# Теоретическая информация о искусственной жизни

## 1.1 Общие сведения

В конце 80-х — начале 90-х годов возникло одно интересное направление кибернетических исследований: «Искусственная жизнь» (англ. Artificial Life или Alife).

Основной мотивацией исследований искусственной жизни служит желание понять и промоделировать формальные принципы организации биологической жизни. Как сказал руководитель первой международной конференции по искусственной жизни К. Лангтон «основное предположение искусственной жизни состоит в том, что «логическая форма» организма может быть отделена от материальной основы его конструкции».

Сторонники направления «Искусственная жизнь» часто считают, что они исследуют более общие формы жизни, чем те, которые существуют на Земле. Т.е. изучается жизнь, какой она могла бы в принципе быть («life-as-it-could-be»), а не обязательно та жизнь, какой мы ее знаем («life-as-we-know-it»).

Искусственная жизнь — это синтетическая биология, которая по аналогии с синтетической химией пытается воспроизвести биологическое поведение в различных средах. Это жизнь, созданная человеком, а не природой («life made by Man rather than by Nature»). Исследования искусственной жизни направлены не только на теоретические исследования свойств жизни, но и (аналогично синтетической химии) на практические приложения, такие как подвижные роботы, медицина, нанотехнология, «жизнь» социальных систем и тому подобное.

Большую роль в исследованиях искусственной жизни играет математическое и компьютерное моделирование. Очень часто «организмы» в искусственной жизни — это придуманные людьми организмы, живущие в мире компьютерных программ.

## Модели искусственной жизни

Модели искусственной жизни – активно развивающаяся область исследований. Большинство моделей – остроумные компьютерные эксперименты. Серьезное математическое описание здесь только начинается.

Исследования искусственной жизни тесно связаны с другими интересными направлениями: моделями происхождения жизни, автоматами С.А. Кауффмана, работами по прикладному эволюционному моделированию, по теории нейронных сетей. Эволюция популяций искусственных организмов – одно из ведущих направлений исследований искусственной жизни. Модели эволюции здесь часто основаны на генетическом алгоритме. Но в моделях искусственной жизни часто не вводится явно понятие приспособленности, как это делается в генетических алгоритмах. Приспособленность проявляется естественным путем: особи рождаются, когда их предки готовы дать потомков, и погибают, когда не хватает пищи или когда их убивает другая особь. В этом случае говорят, что приспособленность эндогенна. Управление поведением искусственных организмов часто моделируется с помощью нейронных сетей. Модели искусственной жизни проливают новый свет на эволюционные явления.

Некоторые примеры характерных исследований искусственной жизни:

- Исследование динамики жизнеподобных структур в клеточных автоматах (К.Лангтон).

- ПолиМир (PolyWorld) Л. Ягера: компьютерная модель искусственных организмов, которые имеют структурированную нейронную сеть, обладают цветовым зрением, могут двигаться, питаться (и увеличивать тем самым свою энергию), могут скрещиваться и бороться друг с другом. Население обычно исчисляется сотнями, поскольку каждый индивидуум довольно сложен, а окружающая среда потребляет значительные компьютерные ресурсы. Было замечено, что после длительной эволюции спонтанно возникает ряд интересных форм поведения, таких как каннибализм, хищники и жертвы, а также мимикрия.

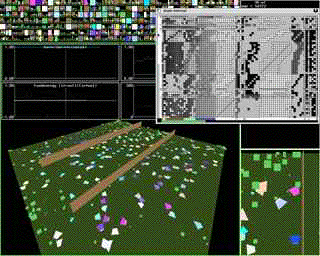


Рисунок 1 – Скриншот работы Polyworld

- Тьерра (Tierra) Т.Рэя: модель эволюции самовоспроизводящися компьютерных программ. «Организмы» Тьерры содержат геномы, которые определяют инструкции исполнительных программ. Есть операционная система, которая определяет: работу с памятью, выделение времени на вычисления каждому существу, подсчёт смертей и рождений, банк успешных геномов, автоматизирует экологический анализ. А программами этой операционной системы являются организмы. Взаимодействия между организмами проводят к эволюционному возникновению сложного «биоразнообразия» самовоспроизводящихся программ.



Рисунок 1 – Скриншот запушенной симуляции Tierra

- Авида (Avida) К. Адами с сотрудниками, эта модель – развитие модели Тьерра. По сравнению с Тьерой Авида более проста и обладает большей общностью. Модель анализировалась аналитическими методами. Были получены характеристики распределения особей в эволюционирующих популяциях. Исследования на модели Авида количественно поддерживают ту точку зрения, что эволюция движется скачками, а не непрерывно.

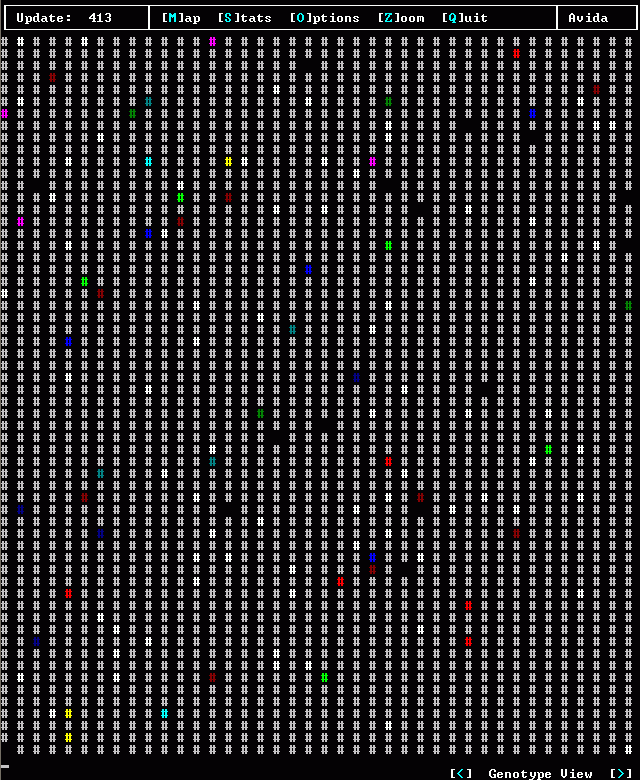


Рисунок 1 – Скриншот работы Avida с открытой вкладкой карты

- Проект “Искусственная жизнь” foo52ru. В этом проекте не происходит прямого программирования алгоритма эволюции или генетического алгоритма, а он возникает автоматически за счёт среды. Используются агенты с программой в виде чисел для программирования поведения. Программа представляет собой ассемблеро-подобный язык, мутация может изменить случайную команду. Агентам необходимо собирать еду и выживать рядом друг с другом и под действием жестокой среды.

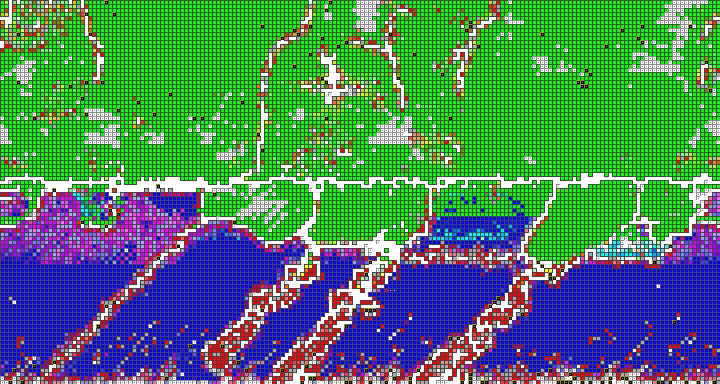


Рисунок 1 - Скриншот работы программы “Искусственная жизнь” foo52ru

- Анализ взаимодействия между обучением и эволюцией, выполненный Д.Экли и М. Литтманом. Эта работа продемонстрировала, что «обучение и эволюция вместе более успешны в формировании адаптивной популяции, чем обучение либо эволюция по отдельности».

- ЭХО (ECHO) Дж.Холланда. Эта модель описывает эволюцию простых агентов, которые взаимодействуют между собой путем скрещивания, борьбы и торговли. Взаимодействия между агентами приводит к формированию различных экологических систем: «войны миров», симбиозов и тому подобное.

- Модель эволюции двух конкурирующих популяций, одна из которых есть популяция программ, решающих определенную прикладную проблему (задачу сортировки), а вторая – популяция задач, эволюционирующих в направлении усложнения проблемы (Д. Хиллис). Первая из популяций может рассматриваться как популяция особей-хозяев, а вторая как популяция паразитов. Моделирование показало, что коэволюция в системе паразит-хозяин проводит к нахождению значительно лучших решений проблемы по сравнению с тем решением, которое можно найти в результате эволюции одной только первой популяции (популяции особей-хозяев).

- «Муравьиная ферма» (AntFarm) Р. Коллинза и Д Джефферсона. Эта модель разработана на базе «Коннекшен-машины» (Connection-Machine). Модель имитирует поведение поиска пищи в огромных эволюционирующих популяциях искусственных муравьев.

- Классифицирующие системы Дж.Холланда с сотрудниками. Это модель эволюции когнитивного процесса. Классифицирующая система есть система индуктивного вывода, которая основана на наборе логических правил. Каждое правило имеет следующую форму: «если <условие>, тогда <действие>». Система правил оптимизируется как посредством обучения, так и эволюционным методом. В процессе обучения меняются приоритеты использования правил (т.е. меняются коэффициенты, характеризующие силу правил). При обучении используется так называемый алгоритм «пожарной бригады»: при успехе поощряются не только те правила, которые непосредственно привели к успешному действию, но и те, которые были предшественниками успеха. Поиск новых правил осуществляется эволюционным методом.

## 1.3 Адаптивное поведение

С начала 90-х годов активно развивается направление "Адаптивное поведение". Основной подход этого направления – конструирование и исследование искусственных (в виде компьютерной программы или робота) "организмов", способных приспосабливаться к внешней среде.

Исследователи направления "Адаптивное поведение" стараются строить такие модели, которые применимы к описанию поведения как реального животного, так и искусственного организма.

Задачи направления "Адаптивное поведение":

– исследовать архитектуры и принципы функционирования, которые позволяют животным или роботам жить и действовать в переменной внешней среде.

– попытаться проанализировать эволюцию когнитивных способностей животных и эволюционное происхождение человеческого интеллекта.

Для исследований "Адаптивного поведения" характерен синтетический подход: здесь конструируются архитектуры, обеспечивающие "интеллектуальное" поведение. Исследователь сам разрабатывает архитектуры, подразумевая, что подобные структуры, обеспечивающие адаптивное поведение, должны быть у реальных организмов.

Направления "Искусственная жизнь" и "Адаптивное поведение" имеют много общего: синтетический подход к конструированию жизнеподобных организмов, попытка промоделировать формальные законы жизни и систем управления, ориентация на компьютерные и математические модели, использование эволюционных концепций и моделей.

Эти направления используют ряд нетривиальных компьютерных методов:

– нейронные сети (англ. Neural Networks)

– классифицирующие системы (англ. Classifier Systems),

– обучение на основе подкрепления (англ. Reinforcement Learning),

– генетический алгоритм и другие методы эволюционной оптимизации

В целом соотношение между направлениями "Искусственная Жизнь" и "Адаптивное Поведение", используемыми в них компьютерными методами, их научным значением и их потенциальными применениями можно представить в виде схемы, показанной на Рис. 1.

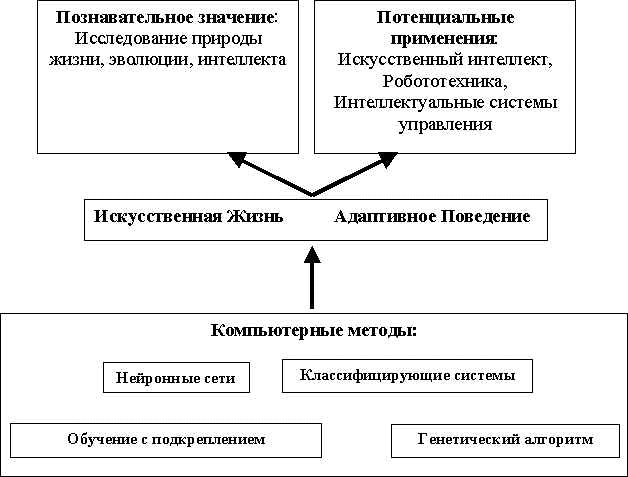


Рисунок 1 – Схема междисциплинарных связей направлений "Искусственная Жизнь" и "Адаптивное Поведение"

Это всё ещё активно развивающиеся направления исследований. По этим направлениям регулярно проводятся международные и европейские конференции "ALIFE (The Conference on Artificial Life)", "ECAL (European Conference on Artificial Life)". Издаётся журнал "Artificial Life".

# Искусственные нейронные сети

Иску́сственные нейро́нные се́ти (ИНС) — математические модели, а также их программные или аппаратные реализации, построенные по принципу организации и функционирования биологических нейронных сетей — сетей нервных клеток живого организма. Это понятие возникло при изучении процессов, протекающих в мозге, и при попытке смоделировать эти процессы. Первой такой попыткой были нейронные сети Маккалока и Питтса. Впоследствии, после разработки алгоритмов обучения, получаемые модели стали использовать в практических целях: в задачах прогнозирования, для распознавания образов, в задачах управления и др.

ИНС представляют собой систему соединённых и взаимодействующих между собой простых процессоров (искусственных нейронов). Такие процессоры обычно довольно просты, особенно в сравнении с процессорами, используемыми в персональных компьютерах. Каждый процессор подобной сети имеет дело только с сигналами, которые он периодически получает, и сигналами, которые он периодически посылает другим процессорам. И тем не менее, будучи соединёнными в достаточно большую сеть с управляемым взаимодействием, такие локально простые процессоры вместе способны выполнять довольно сложные задачи.

С точки зрения машинного обучения, нейронная сеть представляет собой частный случай методов распознавания образов, дискриминантного анализа, методов кластеризации и т. п. С математической точки зрения, обучение нейронных сетей — это многопараметрическая задача нелинейной оптимизации. С точки зрения кибернетики, нейронная сеть используется в задачах адаптивного управления и как алгоритмы для робототехники. С точки зрения развития вычислительной техники и программирования, нейронная сеть — способ решения проблемы эффективного параллелизма. А с точки зрения искусственного интеллекта, ИНС является основой философского течения коннективизма и основным направлением в структурном подходе по изучению возможности построения (моделирования) естественного интеллекта с помощью компьютерных алгоритмов.

Нейронные сети не программируются в привычном смысле этого слова, они обучаются. Возможность обучения — одно из главных преимуществ нейронных сетей перед традиционными алгоритмами. Технически обучение заключается в нахождении коэффициентов связей между нейронами. В процессе обучения нейронная сеть способна выявлять сложные зависимости между входными данными и выходными, а также выполнять обобщение. Это значит, что, в случае успешного обучения, сеть сможет вернуть верный результат на основании данных, которые отсутствовали в обучающей выборке, а также неполных и/или «зашумленных», частично искаженных данных.

## 2.1 Биологический аспект

Нейронные сети возникли из исследований в области искусственного интеллекта, а именно, из попыток воспроизвести способность биологических нервных систем обучаться и исправлять ошибки, моделируя низкоуровневую структуру мозга (Patterson, 1996). Основной областью исследований по искусственному интеллекту в 60-е - 80-е годы были экспертные системы. Такие системы основывались на высокоуровневом моделировании процесса мышления (в частности, на представлении, что процесс нашего мышления построен на манипуляциях с символами). Скоро стало ясно, что подобные системы, хотя и могут принести пользу в некоторых областях, не ухватывают некоторые ключевые аспекты человеческого интеллекта. Согласно одной из точек зрения, причина этого состоит в том, что они не в состоянии воспроизвести структуру мозга. Чтобы создать искусственных интеллект, необходимо построить систему с похожей архитектурой.

Мозг состоит из очень большого числа (приблизительно 10,000,000,000) нейронов, соединенных многочисленными связями (в среднем несколько тысяч связей на один нейрон, однако это число может сильно колебаться). Нейроны - это специальная клетки, способные распространять электрохимические сигналы. Нейрон имеет разветвленную структуру ввода информации (дендриты), ядро и разветвляющийся выход (аксон). Аксоны клетки соединяются с дендритами других клеток с помощью синапсов. При активации нейрон посылает электрохимический сигнал по своему аксону. Через синапсы этот сигнал достигает других нейронов, которые могут в свою очередь активироваться. Нейрон активируется тогда, когда суммарный уровень сигналов, пришедших в его ядро из дендритов, превысит определенный уровень (порог активации).

Интенсивность сигнала, получаемого нейроном сильно зависит от активности синапсов. Каждый синапс имеет протяженность, и специальные химические вещества передают сигнал вдоль него. Один из самых авторитетных исследователей нейросистем, Дональд Хебб, высказал постулат, что обучение заключается в первую очередь в изменениях "силы" синаптических связей. Например, в классическом опыте Павлова, каждый раз непосредственно перед кормлением собаки звонил колокольчик, и собака быстро научилась связывать звонок колокольчика с пищей. Синаптические связи между участками коры головного мозга, ответственными за слух, и слюнными железами усилились, и при возбуждении коры звуком колокольчика у собаки начиналось слюноотделение.

Таким образом, будучи построен из очень большого числа совсем простых элементов (каждый из которых берет взвешенную сумму входных сигналов и в случае, если суммарный вход превышает определенный уровень, передает дальше двоичный сигнал), мозг способен решать чрезвычайно сложные задачи.

## 2.2 Обучение нейронной сети

Процесс функционирования нейросети, то есть сущность действий, которые она способна выполнять, зависит от величин синаптических связей, поэтому, задавшись определенной структурой, отвечающей какой-либо задаче, разработчик сети должен найти оптимальные значения всех переменных весовых коэффициентов (некоторые синаптические связи могут быть постоянными).

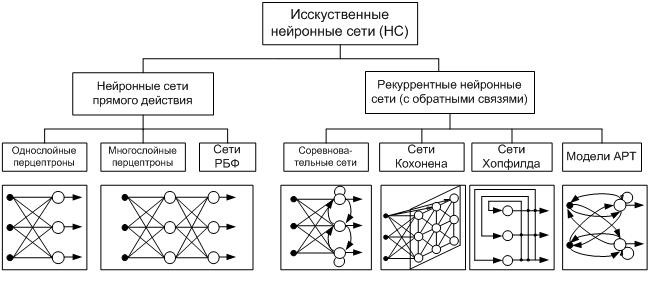
Этот этап называется обучением нейросети, и от того, насколько качественно он будет выполнен, зависит способность сети решать поставленные перед ней проблемы во время эксплуатации. На этапе обучения кроме параметра качества подбора весов важную роль играет время обучения. Как правило, эти два параметра связаны обратной зависимостью и их приходится выбирать на основе компромисса.

Обучение нейронной сети может вестись с учителем или без него. В первом случае сети предъявляются значения как входных, так и желательных выходных сигналов, и она по некоторому внутреннему алгоритму подстраивает веса своих синаптических связей. Во втором случае выходы нейросети формируются самостоятельно, а веса изменяются по алгоритму, учитывающему только входные и производные от них сигналы.

Существует великое множество различных алгоритмов обучения, которые делятся на два больших класса: детерминистские и стохастические. В первом из них подстройка весов представляет собой жесткую последовательность действий, во втором – она производится на основе действий, подчиняющихся некоторому случайному процессу.

## 2.3 Авторство и право владения

ИНС может рассматриваться как направленный граф со взвешенными связями, в котором искусственные нейроны являются узлами. По архитектуре связей ИНС могут быть сгруппированы в два класса: сети прямого распространения, в которых графы не имеют петель, и рекуррентные сети, или сети с обратными связями.

Рис. 1 - архитектуре связей ИНС

В наиболее распространенном семействе сетей первого класса, называемых многослойным перцептроном, нейроны расположены слоями и имеют однонаправленные связи между слоями. На рис. 1 представлены типовые сети каждого класса. Сети прямого распространения являются статическими в том смысле, что на заданный вход они вырабатывают одну совокупность выходных значений, не зависящих от предыдущего состояния сети. Рекуррентные сети являются динамическими, так как в силу обратных связей в них модифицируются входы нейронов, что приводит к изменению состояния сети.

## 2.4 Смарт-контракты

Смарт-контракт – это компьютерный алгоритм, выполняющий что-либо в зависимости от действий другого объекта, предназначен для заключения и поддержания коммерческих контрактов в технологии блокчейн. Стороны подписывают умный контракт, используя аналогичные подписанию отправки средств в действующих криптовалютных сетях методы. После подписания сторонами контракт вступает в силу. Для обеспечения автоматизированного исполнения обязательств контракта непременно требуется среда существования, которая позволяет полностью автоматизировать выполнение пунктов контракта. Это означает, что умные контракты смогут существовать только внутри среды, имеющей беспрепятственный доступ исполняемого кода к объектам умного контракта.

Все условия контракта должны иметь математическое описание и ясную логику исполнения. В связи с этим первые умные контракты имеют задачу формализации наиболее простых взаимоотношений, состоящих из небольшого количества условий. Умные контракты, к примеру, могут следить за выполнением условий долгосрочных кредитов.

## 2.5 Электронные медицинские карты

Блокчейн позволяет обеспечить неизменность информации, проследить происхождение данных и их сохранность, а также использовать смарт-контракты. Все эти характеристики делают блокчейн очень подходящей технологией для хранения и управления электронными медицинскими записями пациентов, а также для обмена такими записями

## 2.6 Медицинские исследования

Блокчейн способен совершить революцию в медицинских исследованиях. Технология может ускорить проведение исследований, открыть доступ к данным, а также усилить контроль за результатами. Блокчейн упрощает предоставление пациентами разрешения на использование их данных для клинических исследований. Кроме информации о согласии пациента, технология позволяет хранить различные типы данных из разных источников: об уходе за пациентами, о проведенных клинических испытаниях, о биомаркерах, о цепочке поставок фармацевтической продукции и другие. Анализ этих разрозненных данных позволяет повысить эффективность клинических и биомедицинских исследований. Также, он способен устранить фальсификацию данных и исключение нежелательных результатов клинических исследований. Свойство неизменяемости блокчейна подтверждает целостность данных, собранных для проведения клинических исследований.

## 2.7 Цепочки поставок и борьба с контрафактом

Другой сценарий применения блокчейна в медицине – управление цепочками поставок, а также борьба с контрафактом. Поддельные и некачественные лекарства – общая проблема фармацевтической отрасли. Глобальный рынок контрафактных лекарств оценивается в $200 млрд. Блокчейн способен решить эту проблему, обеспечивая прозрачность на всех этапах производства, поставки и реализации медицинских препаратов.

## 2.8 Благотворительность

Главной проблемой благотворительности всегда был вопрос доверия. Жертвующие никак не можем отследить дошли ли деньги до нуждающихся или же осели в чьих-то карманах. Блокчейн и даёт эту самую прозрачность, и уверенность в том, что адресат получил пожертвования.

Ещё один из важный плюсов – это возможность в пару кликов поддержать любую организацию вне зависимости от того, где она находится - в России, Австралии или Америке. И количество организаций, принимающих цифровую валюту, постоянно растет от старых добрых фондов до интернет-аукционов.

# Реализация тренажёра

Понять то, как работает технология блокчейна довольно непросто, особенно без предварительной подготовки. Практическая часть исследования состоит в разработке тренажёра для студентов, который позволит разобраться в базовых принципах работы блокчейна. К тренажёру написаны рекомендации по использованию, чтобы студент мог полностью попробовать весь функционал и не тратил время на дополнительный поиск информации и попытки разобраться с тем, как пользоваться тренажёром. Код и рекомендации размещены на веб-сервисе контроля версий GitHub по адресу https://github.com/ggrechka/blockchain-trainer. На рисунке 10 изображено наполнение проекта

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 10 – Наполнение проекта на GitHub

## 3.1 Инструменты для разработки

Для разработки был использован язык Python 3.8, фреймворк Flask 2.1.2., библиотека Requests 2.27.1 для выполнения запросов. Также были использованы:

* модуль time для фиксации времени создания блоков и транзакций;
* git для хранения версий;
* hashlib для хеширования SHA256;
* json для визуализации данных;
* uuid4 для создания случайного уникального идентификатора;
* jsonify для отправки json-файлов;
* urlparse для разбора URL на компоненты.

## 3.2 Требования к реализации

Для студентов критически важным будет понимание базовых принципов работы блокчейн сетей, без углубления в тонкости. Поэтому тренажёр должен:

* содержать все базовые элементы блокчейна (блоки, транзакции и т.д);
* иметь функционал для ручного инициирования процессов, чтобы изучить последовательность и логику работу блокчейна;
* включать в себя имитацию технологии распределённых сетей, расположенную на локальном компьютере.

## 3.3 Описание функциональности тренажёра

Описанный функционал и рекомендуемый алгоритм обучения на тренажёре расположен на GitHub.

Для начала работы необходимо:

* запустить сервер через терминал;
* скачать, установить и открыть приложение Postman.

Для запуска сервера необходимо ввести в терминал следующую команду, заменив ПОРТ на номер желаемого сетевого порта: pipenv run python main.py --port ПОРТ.

Проверив, что, после запуска сервера, отправленный GET запрос на localhost:ПОРТ/ping возвращает код 200 OK, можно приступать к изучению тренажёра. На рисунке 11 показано, где искать подтверждение корректной работе сервера.

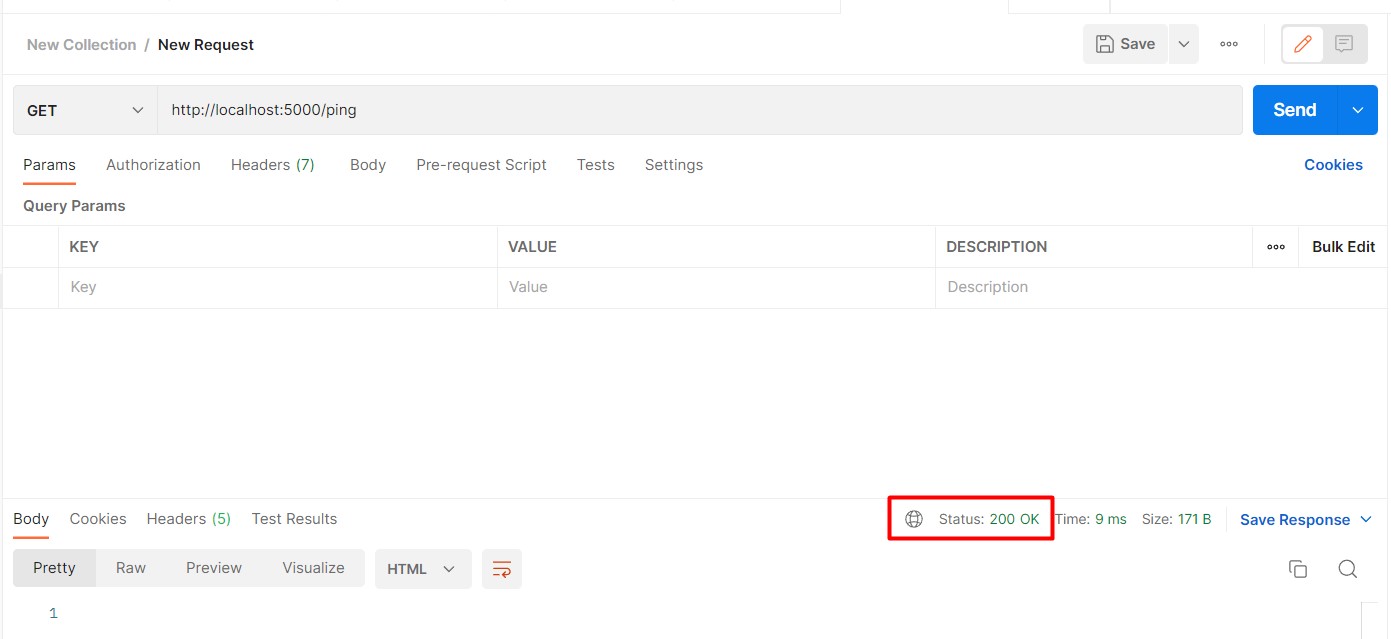


Рисунок 11 – Окно с корректным запуском сервера

Весь тренажёр построен на распределённой сети, которая реализуется через локальные порты компьютера. Для примера представим, что сейчас в сети существует 2 узла с портами 5000 и 5002. Чтобы они корректно работали, их нужно зарегистрировать в блокчейне. Необходимо отправить POST-запрос на http://localhost:5000/nodes/register с телом запроса во вкладке Body, который выглядит так, как изображено на рисунке 12.

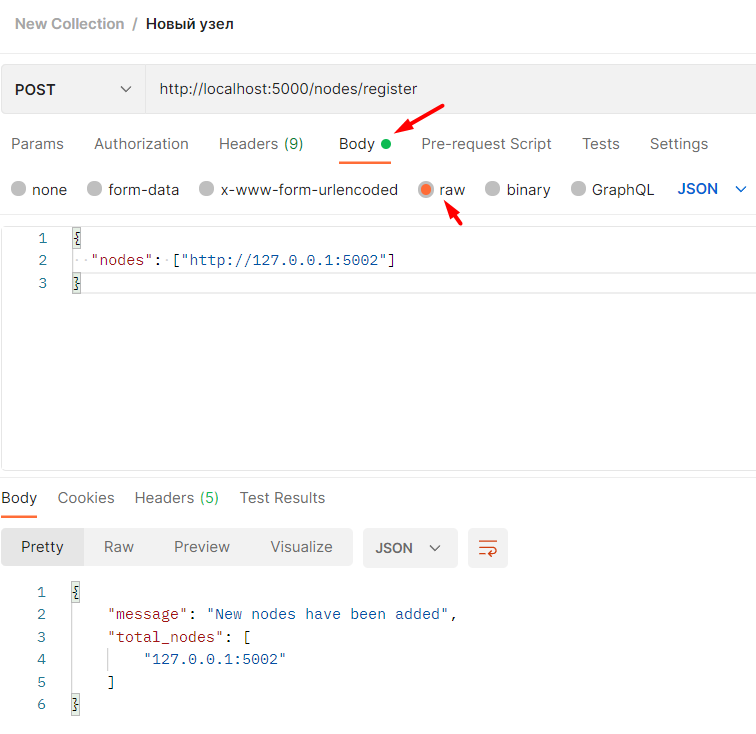


Рисунок 12 – Окно с успешным добавлением одного из узлов

Теперь в блокчейне существует 2 узла с портами 5000 и 5002. Чтобы намайнить новый блок, необходимо с любого порта отправить GET-запрос на http://localhost:5002/mine. Ответ на запрос показан на рисунке 13. Программа намайнила новый блок, в котором есть первая транзакция, представляющая собой оплату вознаграждения майнеру.

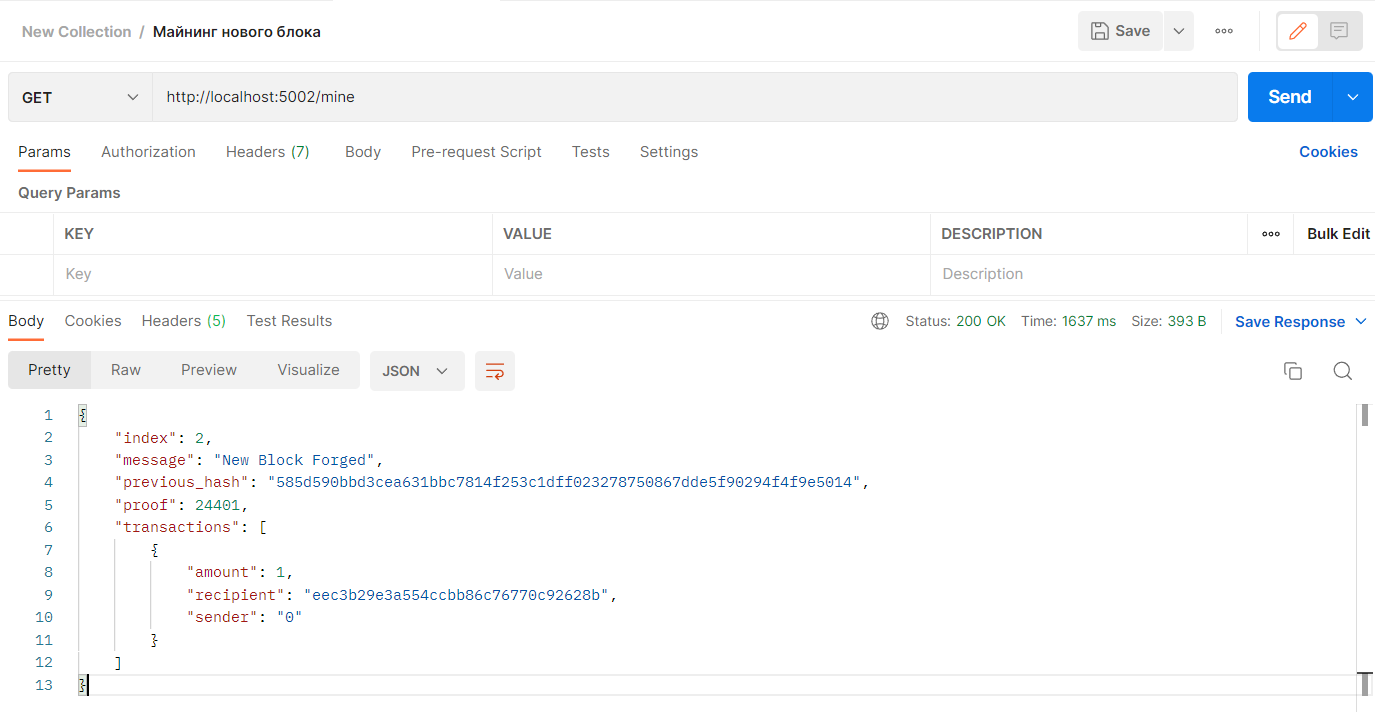


Рисунок 13 – Окно с намайненным блоком

После майнинга нового блока, следует перейти к созданию транзакций. Для этого нужно отправить POST-запрос на http://localhost:5002/transactions/new с телом запроса, как на рисунке 14.

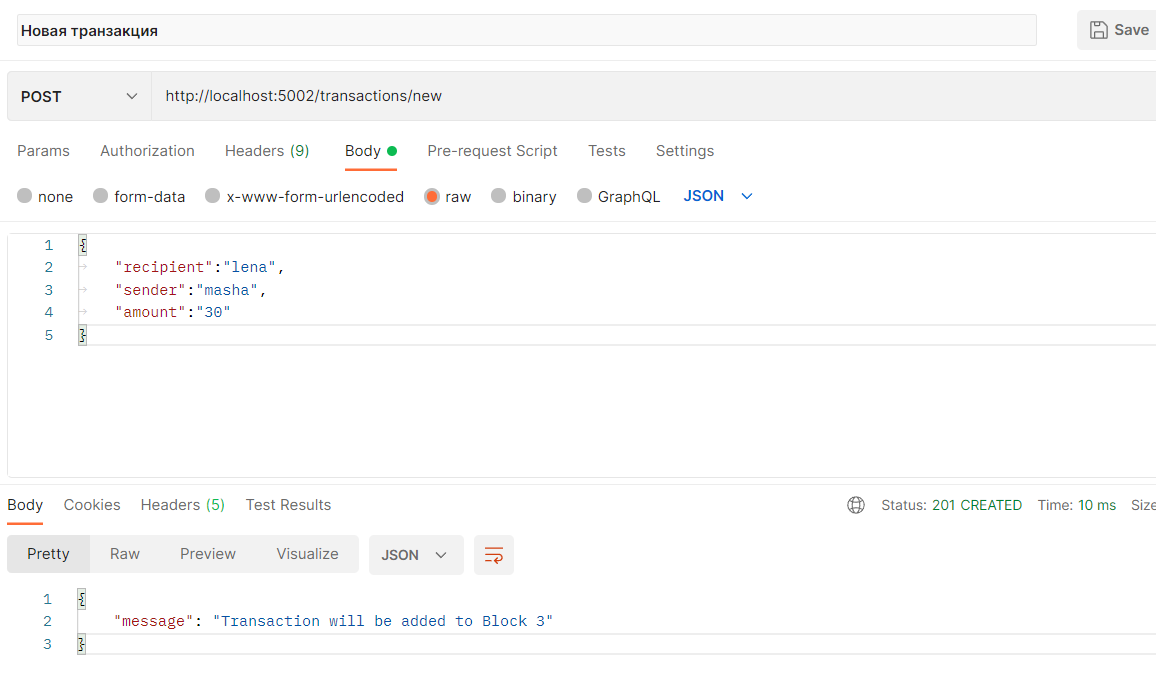


Рисунок 14 – Окно с успешной новой транзакцией

В этой транзакции Маша отправила Лене 30 монеток. В настоящих блокчейнах нет никаких имён, а указаны адреса, но в данной ситуации это не критично для понимания работы блокчейна.

Продолжая создавать транзакции, можно снова отправить GET-запроса на http://localhost:5002/chain и посмотреть, что сейчас есть в цепочке. Результат запроса изображён на рисунке 15.

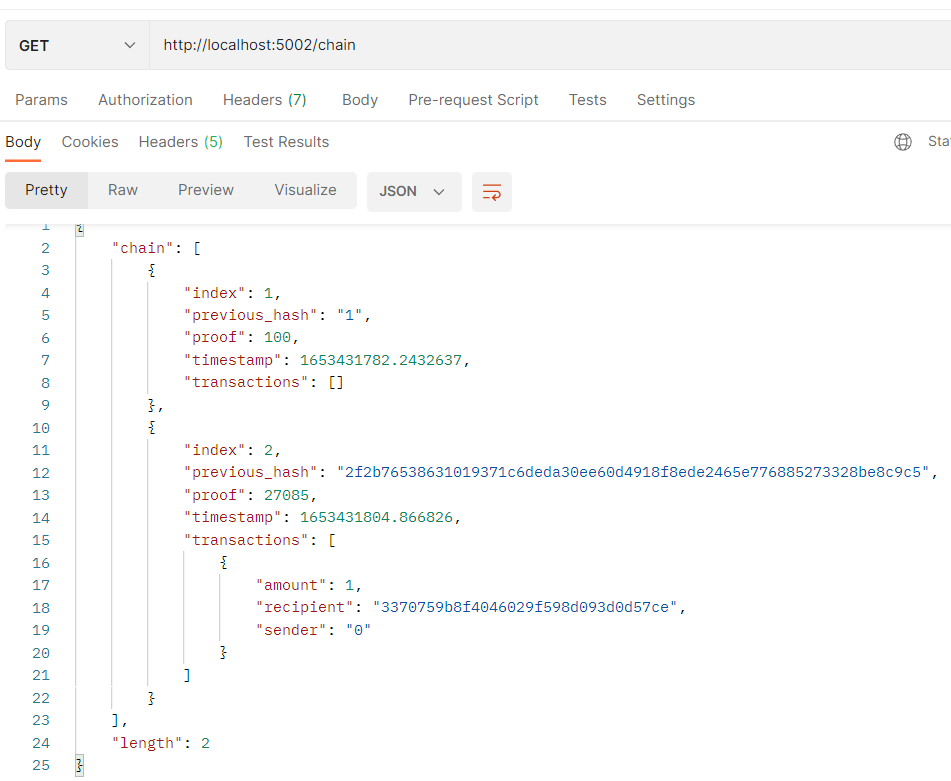


Рисунок 15 – Окно со списком всей цепочки у узла 5002

Как и ожидалось, отправленной транзакции от Маши к Лене сейчас нет в цепочке у узла 5002, так как эта транзакция находится в списке необработанных транзакций. Чтобы она появилась, необходимо намайнить новый блок, который соберёт в себе все необработанные транзакции. На рисунке 16 показана цепочка после майнинга нового блока. Транзакция от Маши к Лене была проведена.

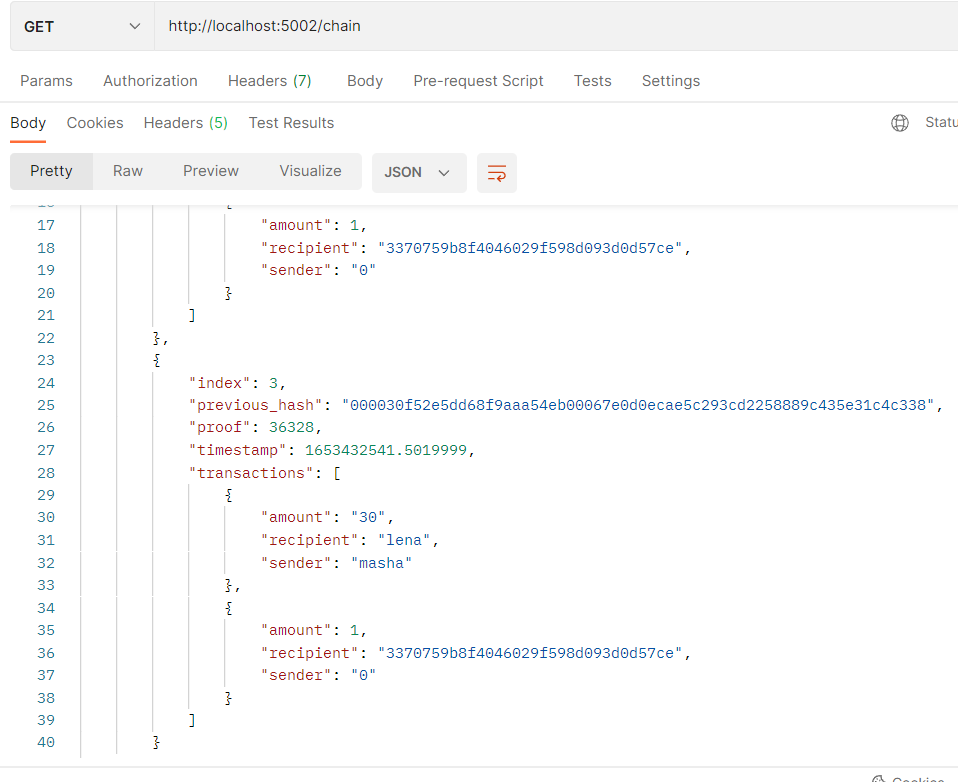


Рисунок 16 – Окно со списком всей цепочки у узла 5002 после майнинга

Необходимо отметить, что все перечисленные выше запросы были отправлены с узла 5002. Если посмотреть на цепочку у узла 5000 (рисунок 17) можно увидеть отсутствие новых блоков и транзакций.

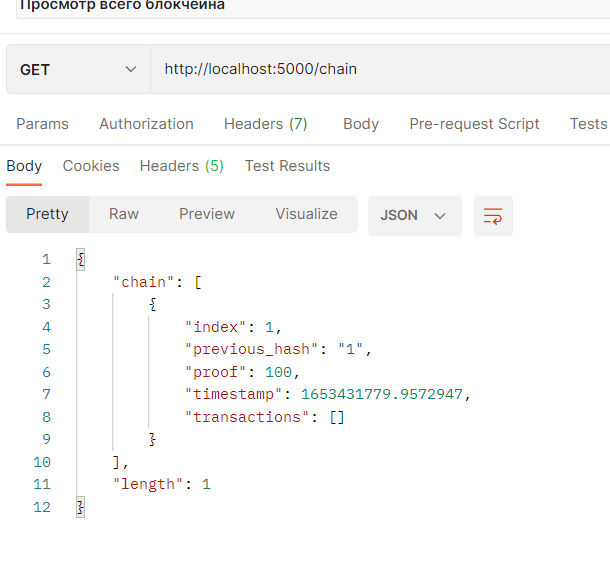


Рисунок 17 – Окно со списком всей цепочки у узла 5000

Таких ситуаций в реальных блокчейнах не бывает, так как все процессы происходят автоматически, в этом тренажёре студенту предлагается инициировать процессы самостоятельно, чтобы понять логику работы блокчейна.

Чтобы узел 5000 узнал обо всех блоках узла 5002, необходимо применить алгоритм консенсуса. В нашей ситуации это самый известный Proof Of Work. На рисунке 18 изображён ответ GET-запроса на http://localhost:5000/nodes/resolve.

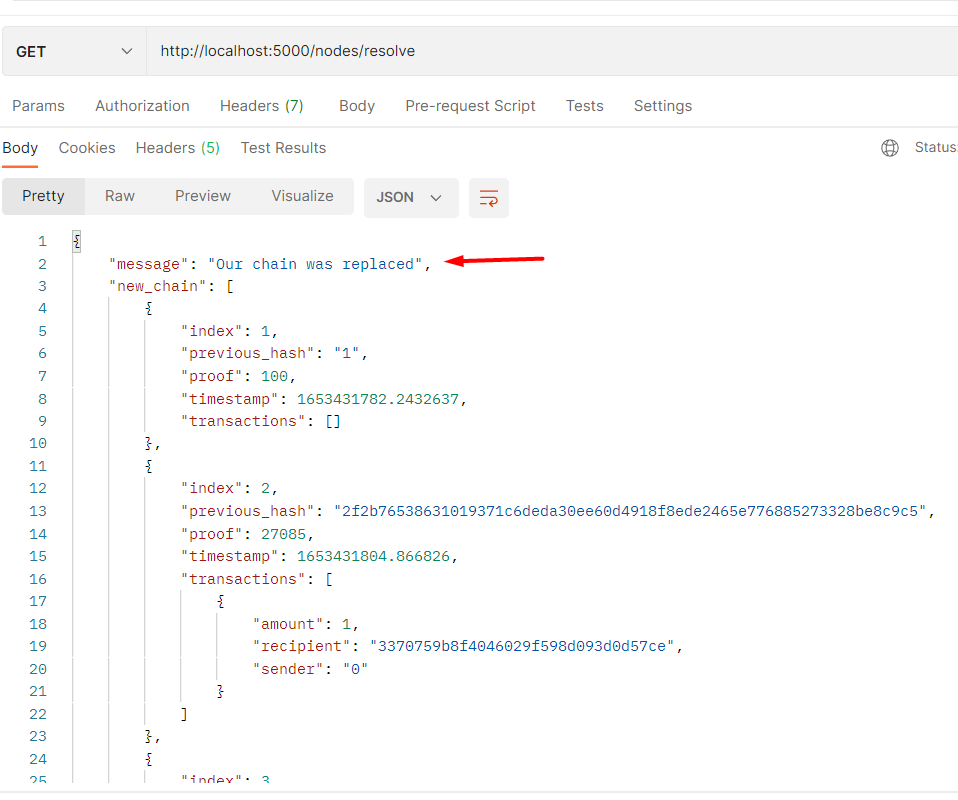


Рисунок 18 – Окно после применения алгоритма консенсуса

После применения алгоритма консенсуса, цепочки у узла 5000 пополнились всеми блоками, которые создал узел 5002. Для наглядности можно запустить алгоритм и на узле 5002, результат можно увидеть на рисунке 19.

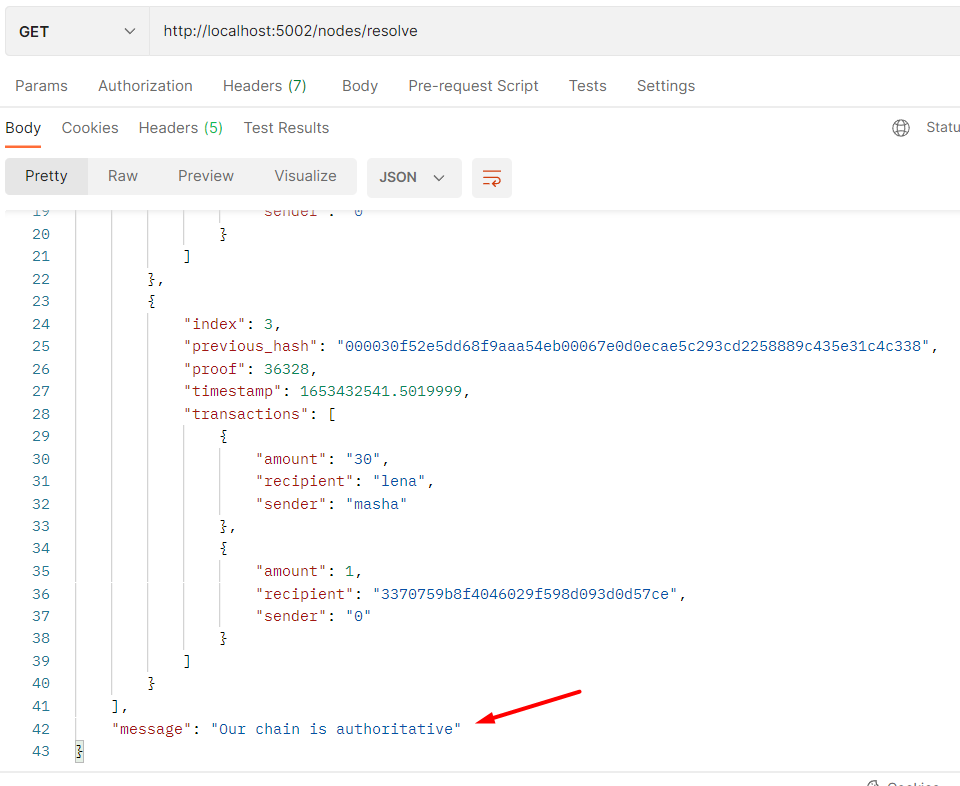


Рисунок 19 – Окно после применения алгоритма консенсуса

Алгоритм ничего не менял, а лишь подтвердил, что цепочка у узла 5002 сейчас самая актуальная.

# ЗАКЛЮЧЕНИЕ

В работе были изучены основные принципы работы блокчейн сетей, а также рассмотрены самые актуальные сферы их применения. Были подробно изучены возможности языка программирования Python и его фреймворка Flask. Рассмотрена работа с HTTP-запросами с помощью библиотеки Request.

По результатам исследования был разработан тренажёр блокчейна для студентов, который поможет им на практике разобраться в базовых понятиях и логике работы блокчейн сетей.

В дальнейшем планируется углубленное изучение криптовалют, построенных на блокчейне, платформы Ethereum, а также погружение в систему работы смарт-контрактов.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Python 3.8.12 Documentation: — Текст: электронный — 2022. — URL: https://docs.python.org/3.8/ (дата обращения 20.05.2022).

2 Welcome to Flask — Flask Documentation (2.1.x) — Текст: электронный — 2010. — URL: https://flask.palletsprojects.com/en/2.1.x/ (дата обращения 21.05.2022).

3 Блокчейн: как он работает, и почему эта технология изменит мир — Текст: электронный — 2017. — URL: https://habr.com/ru/company/iticapital/blog/340992/ (дата обращения 12.12.2021).

4 Что нам стоит блокчейн построить? — Текст: электронный — 2019. — URL: https://habr.com/ru/post/443282/ (дата обращения 14.12.2021).

5 Башир И. Блокчейн: архитектура, криптовалюты, инструменты разработки, смарт-контракты; [пер. с англ. М.А.Райтман]. — Москва: ДМК Пресс, 2019. — 538, с.: ил. — ISBN 978-5-97060-624-7.

# ПРИЛОЖЕНИЕ А

**Основная программа**

Файл main.py

import hashlib

import json

from time import time

from urllib.parse import urlparse

from uuid import uuid4

import requests

from flask import Flask, jsonify, request

class Blockchain:

def \_\_init\_\_(self):

self.current\_transactions = [] # список для хранения транзакций

self.chain = [] # список для хранения всех блоков

self.nodes = set() # список узлов

# Генезис блок - самый первый блок без предыдущего

genesis = self.new\_block(previous\_hash='1', proof=100)

self.chain = [genesis]

def register\_node(self, address):

parsed\_url = urlparse(address)

if parsed\_url.netloc:

self.nodes.add(parsed\_url.netloc)

elif parsed\_url.path:

self.nodes.add(parsed\_url.path)

else:

raise ValueError('Invalid URL')

def valid\_chain(self, chain):

"""

Проверяем, является ли внесенный в блок хеш корректным

chain(list) - blockchain

возвращаем булевую переменную: True или False

"""

last\_block = chain[0]

current\_index = 1

while current\_index < len(chain):

block = chain[current\_index]

print(f'{last\_block}')

print(f'{block}')

print("\n-----------\n")

if block['previous\_hash'] != self.hash(last\_block):

return False

if not self.valid\_proof(block):

return False

last\_block = block

current\_index += 1

return True

def resolve\_conflicts(self):

"""

Это алгоритм Консенсуса, он разрешает конфликты,

заменяя нашу цепь на самую длинную в цепи

Возвращает булевую переменную

"""

neighbours = self.nodes

new\_chain = None

# Ищем только цепи, длиннее нашей

max\_length = len(self.chain)

# Захватываем и проверяем все цепи из всех узлов сети

for node in neighbours:

response = requests.get(f'http://{node}/chain')

print(response)

if response.status\_code == 200:

length = response.json()['length']

chain = response.json()['chain']

# Проверяем, является ли длина самой длинной, а цепь - валидной

if length > max\_length and self.valid\_chain(chain):

max\_length = length

new\_chain = chain

# Заменяем нашу цепь, если найдем другую валидную и более длинную

if new\_chain:

self.chain = new\_chain

return True

return False

def new\_block(self, proof, previous\_hash):

"""

Создание нового блока

proof(integer) - Доказательства проведенной работы

previous\_hash - Хеш предыдущего блока

Возвращаем новый блок

"""

block = {

'index': len(self.chain) + 1,

'timestamp': time(),

'transactions': self.current\_transactions,

'proof': proof,

'previous\_hash': previous\_hash or self.hash(self.chain[-1]),

}

self.current\_transactions = []

return block

# внесение новой транзакции

def new\_transaction(self, sender, recipient, amount):

"""

Вносим новую транзакцию

sender - Адрес отправителя

recipient(string) - Адрес получателя

amount(integer) - Сумма

Возвращаем индекс блока, который будет хранить эту транзакцию

"""

self.current\_transactions.append({

'sender': sender,

'recipient': recipient,

'amount': amount,

})

return self.last\_block['index'] + 1

@property

# Декоратор property — это функция, которая принимает другую функцию в качестве аргумента

# и возвращает ещё одну функцию

def last\_block(self):

return self.chain[-1]

@staticmethod

# staticmethod используется для преобразования функции в статическую функцию.

# Статический метод-это метод, который принадлежит классу, а не экземпляру класса

# Статические методы не требуют создания экземпляра

def hash(block):

# Создает хэш SHA-256 блока

block\_string = json.dumps(block, sort\_keys=True).encode()

return hashlib.sha256(block\_string).hexdigest()

# майнинг

def proof\_of\_work(self, current\_block):

while self.valid\_proof(current\_block) is False:

current\_block["proof"] += 1

return current\_block

def valid\_proof(self, block):

guess = self.hash(block)

return guess[:4] == "0000"

app = Flask(\_\_name\_\_)

node\_identifier = str(uuid4()).replace('-', '')

blockchain = Blockchain()

# майнинг нового блока

@app.route('/mine', methods=['GET'])

def mine():

# if port == "5000":

# return jsonify({'message': 'You dont have rules :('}), 200

# else:

last\_block = blockchain.last\_block

blockchain.new\_transaction(

sender="0",

recipient=node\_identifier,

amount=1,

)

previous\_hash = blockchain.hash(last\_block)

block = blockchain.new\_block(0, previous\_hash)

proofed = blockchain.proof\_of\_work(block)

blockchain.chain.append(proofed)

response = {

'message': "New Block Forged",

'index': proofed['index'],

'transactions': proofed['transactions'],

'proof': proofed['proof'],

'previous\_hash': proofed['previous\_hash'],

}

return jsonify(response), 200

# создание новой транзакции в блоке

@app.route('/transactions/new', methods=['POST'])

def new\_transaction():

values = request.get\_json()

required = ['sender', 'recipient', 'amount']

if not all(k in values for k in required):

return 'Missing values', 400

index = blockchain.new\_transaction(values['sender'], values['recipient'], values['amount'])

response = {'message': f'Transaction will be added to Block {index}'}

return jsonify(response), 201

# вывод всех цепочек

@app.route('/chain', methods=['GET'])

def full\_chain():

response = {

'chain': blockchain.chain,

'length': len(blockchain.chain),

}

return jsonify(response), 200

# принятие список новых узлов в форме URL-ов

@app.route('/nodes/register', methods=['POST'])

def register\_nodes():

values = request.get\_json()

nodes = values.get('nodes')

if nodes is None:

return "Error: Please supply a valid list of nodes", 400

for node in nodes:

blockchain.register\_node(node)

response = {

'message': 'New nodes have been added',

'total\_nodes': list(blockchain.nodes),

}

return jsonify(response), 201

# реализации алгоритма Консенсуса, который решает любые конфликты, связанные с подтверждением того,

# что узел находиться в своей цепи

@app.route('/nodes/resolve', methods=['GET'])

def consensus():

replaced = blockchain.resolve\_conflicts()

if replaced:

response = {

'message': 'Our chain was replaced',

'new\_chain': blockchain.chain

}

else:

response = {

'message': 'Our chain is authoritative',

'chain': blockchain.chain

}

return jsonify(response), 200

@app.route('/ping', methods=['GET'])

def ping():

return "", 200

if \_\_name\_\_ == '\_\_main\_\_':

from argparse import ArgumentParser

parser = ArgumentParser()

parser.add\_argument('-p', '--port', default=5000, type=int, help='port to listen on')

args = parser.parse\_args()

port = args.port

app.run(host='127.0.0.1', port=port)