МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Забайкальский государственный университет»

(ФГБОУ ВО «ЗабГУ»)

Факультет энергетический

Кафедра информатики, вычислительной техники и прикладной математики

**ОТЧЕТ**

по производственной практике (научно-исследовательская работа)

(вид практики)

на (в) \_ ООО «Агенство Джин»\_

(место прохождения практики, указать полное наименование организации)

обучающегося Лягоцкого Максима Владимировича

(имя, отчество, фамилия)

Курс\_\_4\_\_ Группа \_\_ИВТ-19\_

Направление подготовки: 09.03.01 Информатика и вычислительная техника

(код, наименование)

Направленность ОП Программное обеспечение вычислительной техники и автоматизированных систем

Руководитель практики от университета:

к. ф. – м. н., доцент, доцент кафедры ИВТ и ПМ Коган Евгения Семёновна

(должность) (подпись) (имя, отчество, фамилия)

Руководитель практики от предприятия:

директор Деревцов Андрей Владимирович

(должность) (подпись) (имя, отчество, фамилия)

(печать организации)

Чита

2023

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Забайкальский государственный университет»

(ФГБОУ ВО «ЗабГУ»)

Факультет энергетический

Кафедра информатики, вычислительной техники и прикладной математики

ЗАДАНИЕ

на производственную практику (научно-исследовательская работа)

Студенту: Лягоцкому Максиму Владимировичу

Направление подготовки: 09.03.01 Информатика и вычислительная техника

Индивидуальное задание на практику:

1. Изучение основных направлений деятельности организации. Формирование требований к программному средству.
2. Анализ данных
3. Разработка и реализация основных алгоритмов, обеспечивающих логику приложения
4. Разработка и реализация логики приложения
5. Написание и защита отчета по практике.

Дата выдачи задания 24.04.2023 г.

Руководитель от университета \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Е.С. Коган

(подпись, расшифровка подписи)

Задание принял к исполнению 24.04.2023 г.

Подпись студента \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/М. В. Лягоцкий/

(И.О.Ф.)

СОДЕРЖАНИЕ

[Введение 4](#_Toc136641698)

[1 Постановка и анализ задачи 5](#_Toc136641699)

[1.1 Описание предметной области 5](#_Toc136641700)

[1.2 Постановка задачи 8](#_Toc136641701)

[1.3 Обоснование выбора средств реализации 10](#_Toc136641702)

[2 Анализ данных 10](#_Toc136641703)

[3 Программная реализация 10](#_Toc136641704)

[3.1 Расписание и события. 10](#_Toc136641705)

[3.2 Генерация рекомендаций 11](#_Toc136641706)

[3.3 Синхронизация опыта пользователей 12](#_Toc136641707)

[3.4 Чек-лист 12](#_Toc136641708)

# ВВЕДЕНИЕ

В современном мире человеку приходится решать все больше задач, обрабатывать огромное количество информации и стараться оставаться бодрыми и жизнерадостными в этих условиях. На помощь людям приходят приложения. Разнообразные ежедневники и чек-листы, буквально захватили просторы магазинов приложений и насчитывают миллионы скачиваний.

Но, не смотря на обилие вариантов на рынке, тяжело подобрать приложение, подходящее под конкретного пользователя. Кого-то не устраивает интерфейс, других перегруженность функционала или его скудность и, наконец, непонимание, как пользоваться приложением, что бы достичь желаемых результатов.

Реализованное в рамках работы приложение направленно не только стать удобным инструментом для помощи в управлении временем пользователя, но и направленно помочь пользователю научиться распределять свои задачи, опираясь на опыт других пользователей.

# 1 Постановка и анализ задачи

## 1.1 Описание предметной области

Первое, что необходимо рассмотреть, говоря о задачах тайм-менеджмента – это психологическая основа приложения. Все люди индивидуальны и нужно определить набор параметров, которыми можно будет описать любого пользователя.

Для этого был разработан тест, состоящий из двенадцати вопросов, позволяющих разделить всех пользователей на кластеры.[1] Это позволяет точнее учесть индивидуальные потребности пользователей.

Перед разработкой приложения был проведен анализ теста, сто прошедших его пользователей были разделены на кластеры. Кластеризация была проведена несколькими методами, такими как k-means, алгоритм минимального покрывающего древа и иерархическая кластеризация.[2] Последний показал наиболее четкие понятные результаты, подтвердив, что тест позволяет удачно разделить пользователей на кластеры. Дендрограмма на рисунке 1.1.1 наиболее точно демонстрирует результат кластеризации.

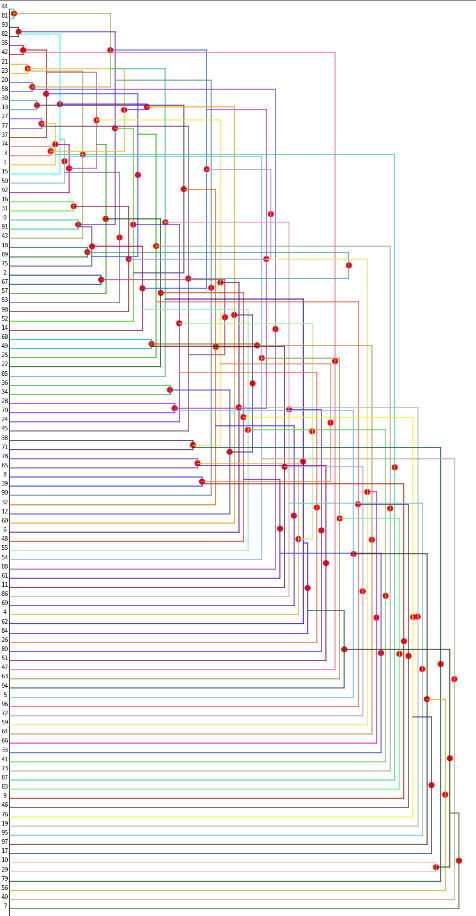


Рисунок 1.1.1 – Дендрограмма

Основная идея приложения и его главное преимущество над своими аналогами – нейронная сеть, способная давать пользователю рекомендации.

При работе с нейронной сетью для обучения модели может использоваться известная целевая переменная (задачи такого типа называются «обучение с учителем»), либо модель самостоятельно учится находить закономерности с имеющихся данных, не имея заранее известные правильные результаты (такой тип задач называется «обучение без учителя»).[3] Обучение с подкреплением (Reinforcement Learning, RL) не относится ни к первому типу, ни ко второму, однако обладает свойствами и того, и другого.

Базовым методом RL является Q-learning. В этом методе используется Q-функция, которая каждому действию ставит в соответствие возможную награду за его совершение в этом конкретном состоянии. Проще всего представить Q-функцию как таблицу, где строками являются все возможные состояния, столбцами все возможные действия, а в ячейках хранится информация о том, какую награду принесет это действие в этом состоянии.

Для учета опыта других пользователей, нейронная сеть сохраняет каждую сгенерированную рекомендацию в буфер. После чего этот опыт используется для дальнейшего обучения сети. Данная схема называется Experience Replay.[4]

В классическом Experience Replay выбор каждой траектории из буфера памяти равновероятен. Однако, не все действия одинаково полезны для агента. К записям в буфере можно добавить веса, которые будут показывать насколько ценным для обучения нейросети был этот опыт. Чем больше вес траектории, тем больше вероятность её выбора для обучения. Это позволяет чаще проигрывать полезный опыт и меньше выбирать бесполезный.

Агент собирает действия в буфер и независимо от этого обучается. Соответственно, задачу наполнения буфера и задачу обучения нейросети можно отделить друг от друга. Для этого была предложена архитектура Ape-x. Нейросеть, используемую как функцию оценки ценности действия, назвали Learner, а агента, выполняющего действие в среде, назвали Actor. Очевидно, Actor может быть не один. Каждый Actor имеет свой собственный экземпляр окружающей среды, что позволяет накапливать опыт гораздо быстрее. Однако, все полученные траектории записываются в один общий буфер Replay, по опыту из которого учится нейросеть.

Experience replay хоть и хранит прошлые действия, но используется только для обучения нейросети. Решение о действии в состоянии все ещё принимается только на основе наблюдаемого состояния, без учета предыдущих. Для устранения этого недочета в проекте используется архитектура R2D2 (Recurrent Replay Distributed DQN). При таком подходе мы проверяем, нет ли в буфере подобного опыта и, при наличии, принятие решения основывается на имеющемся опыте (рисунок 1.1.2).

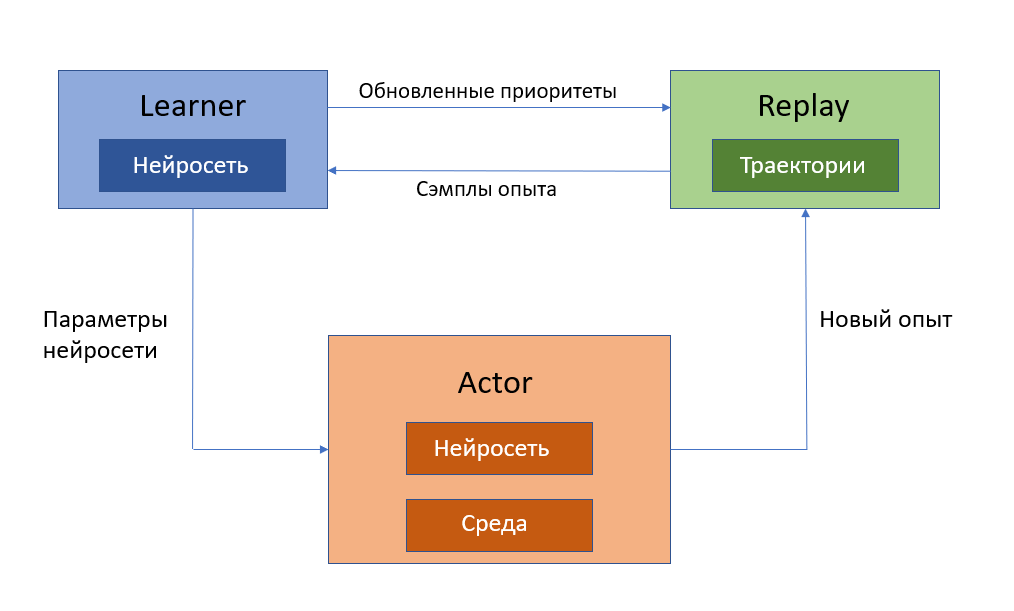


Рисунок 1.1.2 – Архитектура R2D2

## 1.2 Постановка задачи

Целью работы является создание удобного мобильного приложения, способного помочь пользователю распоряжаться своим временем так, что бы его жизнь становилась более комфортной.

Ниже представлен рисунок, представляющий диаграмму вариантов использования разрабатываемого приложения (рисунок 1.2.1)

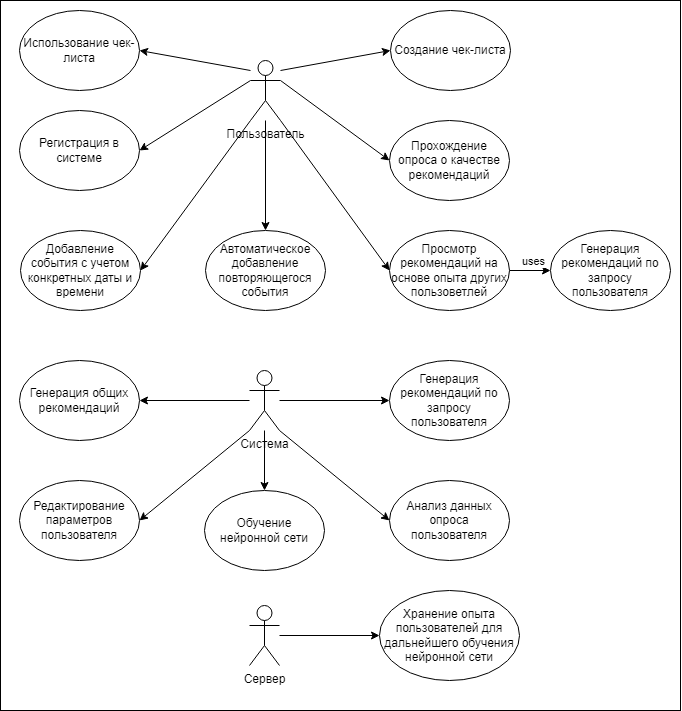


Рисунок 1.2.1 – Диаграмма вариантов использования

## 1.3 Обоснование выбора средств реализации

Для реализации данного приложения был выбран язык Python, так как он более удобен при работе с анализом и прогнозированием. А так же существует множество эффективных библиотек для реализации необходимого функционала.

В качестве IDE был использован PyCharm. Данная среда позволяет предоставляет удобный интерфейс для работы с большими проектами на Python, давая подсказки и отмечая ошибки в коде.

# 2 Анализ данных

Приложение работает с несколькими типами данных, из них стоит выделить:

1. входные данные: данные полученные при регистрации в система; результаты опроса о качестве рекомендации; результаты еженедельного опроса о качестве рекомендаций.
2. промежуточные данные: наборы тегов пользователей.
3. выходные данные: сгенерированные рекомендации.

# 3 Программная реализация

Разрабатываемое приложение реализует несколько основных функций. Рассмотрим каждое в отдельности.

## 3.1 Расписание и события.

В первую очередь приложение является ежедневником. Мы можем выбрать конкретный день на главном экране или через специальный календарь. Экран дня представляет собой отсортированный по времени список всех событий, которые запланированы на выбранную дату.

Каждая ячейка с событием содержит его название, время проведения и набор тегов, описывающих событие. Теги выбираются пользователем при создании мероприятия. Такой подход помогает пользователю быстро вспомнить, что представляет собой запланированное мероприятие, а так же позволяет системе понять тип мероприятия.

Для добавления события в расписание существует отдельное окно, где пользователь вводит название мероприятия, выбирает теги, описывающие мероприятие и выбирает дату и время проведения события.

Помимо ввода конкретной даты, ее можно сгенерировать. Подробнее этот процесс описан в следующем разделе.

## 3.2 Генерация рекомендаций

Рекомендации будут выдаваться только на три дня вперед, но анализируемый период – неделя. Фактически вся неделя представляется как наборы тегов, распределенных по промежуткам. Минимальная длина промежутка – 45 минут. Подробнее данное решение изображено на рисунке 3.2.1.

При этом, событие может начинаться в любое время и его теги будут записаны во все промежутки, в которые оно попадает.

После того, как рекомендованное событие прошло, пользователь получает предложение оценить рекомендацию. Эта оценка добавляется к записи опыта. Так же, пользователь раз в неделю будет получать предложение пройти более подробный опрос о качестве рекомендаций. На основе результатов этих опросов – обучается нейронная сеть.[5]

Если пользователь запросил рекомендацию, а похожего опыта в банке еще нет, то приложение, вместо случайного выбора, который предусматривается такой архитектурой, предложит рекомендацию, которая сбалансирует распределение тегов на неделе.

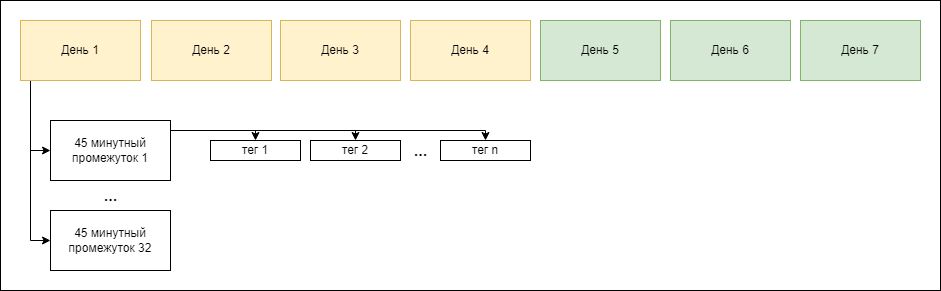


Рисунок 3.2.1 – Схема набора данных

## 3.3 Синхронизация опыта пользователей

Для того, что бы система могла накапливать опыт разных пользователей, реализован сервер, который выступает в роли хранилища данных.

Данные на сервере хранятся не в виде единой базы, а виде отдельных файлов для каждого кластера пользователей. Это позволяет оптимизировать и распараллелить его работу.

Второй функциональной особенностью сервера является очистка от мусора, если какой-то опыт перестает приносить пользу, о чем говорит низкий приоритет опыта, то он удаляется.

При запуске приложения, отправляется запрос на сервер, для загрузки актуальной базы данных для кластера пользователя. Такая система реализована, чтобы пользователю не приходилось каждый раз отправлять запрос на сервер и дожидаться ответа, когда необходимо запросить рекомендацию.

При выходе из приложения, локальный срез данных отправляется на сервер, где опыт встраивается в имеющуюся базу. Если был изменен уже существующий опыт, то сохраняется среднее между параметрами имеющейся и загружаемой записи.

## 3.4 Чек-лист

Последней из основных функций приложения является чек-лист. Фактически, это отдельное окно, где отображаются созданные пользователем чек-листы и их прогресс.

При создании чек-листа, пользователь указывает его название, после чего добавляет необходимое количество подзадач.

При выборе конкретного чек-листа, открывается окно, где перечислены все подзадачи. Пользователь может отметить задачу как выполненную, и она засчитается в прогресс.

Таким образом реализован простой, удобный и минималистичный список дел.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ:

1. Забелин А.А. Дискретная математика; методы и модели теории графов и программная реализация: [Электронный ресурс] / А.А. Забелин, Е.С. Коган – Чита: Наука 2020. URL: [https://reader.lanbook.com/book/173636#81](https://reader.lanbook.com/book/173636%2381). (Дата обращения 16.09.2022)
2. Алгоритмы Краскала, Прима для нахождения минимального остовного древа [Электронный ресурс] URL: <https://habr.com/ru/post/569444/> (Дата обращения 16.09.2022)
3. Нейронные сети для начинающих [Электронный ресурс] URL: <https://habr.com/ru/post/312450/> (Дата обращения 16.07.2022)
4. Использование рекуррентных нейронных сетей в Reinforcement Learning. [Электронный ресурс] URL: <https://habr.com/ru/articles/599923> (дата обращения 01.02.2023).
5. Guide to Reinforcement Learning with Python and TensorFlow [Электронный ресурс] URL: <https://rubikscode.net/2021/07/13/deep-q-learning-with-python-and-tensorflow-2-0/> (дата обращения 01.03.2023).