**Министерство науки и высшего образования**

**Российской Федерации**

**ФЕДЕРАЛЬНОЕ Государственное Автономное**

**образовательное учреждение высшего образования**

**«новосибирский национальный исследовательский государственный университет»**

**Факультет информационных технологий**

Кафедра общей информатики

Направление подготовки 09.03.01 Информатика и вычислительная техника

Направленность (профиль) Программная инженерия и компьютерные науки

**ОТЧЕТ**

**о прохождении производственной практики (преддипломной практики)**

(указывается наименование практики)

**Обучающегося** Кондренко Кирилла Павловича **группы №** 21202 **курса 4**

(Ф.И.О. полностью)

**Тема задания**: Реализация серверной части системы для анализа и индивидуального подбора косметических средств

**Место прохождения практики:** ФГБУН Институт математики им. С. Л. Соболева СО РАН, Лаборатория теории вычислимости и прикладной логики, 630090, Новосибирская обл., Новосибирск, пр. Академика Коптюга, 4

(полное наименование организации и структурного подразделения, индекс, адрес)

**Сроки прохождения практики:** с 03.02.2025 г. по 03.05.2025 г.

**Руководитель практики   
от профильной организации** Пальчунов Дмитрий Евгеньевич, в.н.с.

(Ф.И.О. полностью, должность) (подпись)

**Руководитель практики от НГУ** Яхъяева Гульнара Эркиновна, доцент

(Ф.И.О. полностью, должность) (подпись)

**Руководитель ВКР** Ряскин Александр Николаевич доцент

(Ф.И.О. полностью) (должность)

**Оценка по итогам защиты отчета:**

(неудовлетворительно, удовлетворительно, хорошо, отлично)

**Отчет заслушан на заседании кафедры** общей информатики

(наименование кафедры)

**протокол \_\_\_\_\_\_\_\_\_от** «\_\_\_\_\_\_» мая 2025 г.

Новосибирск 2025

**СОДЕРЖАНИЕ**

[Введение 3](#_Toc197449190)

[Основная часть 5](#_Toc197449191)

[1. Модуль подбора косметических средств 5](#_Toc197449192)

[1.1 Реализация алгоритма 5](#_Toc197449193)

[1.2 Подбор настраиваемых параметров алгоритма 6](#_Toc197449194)

[2. Контейнеризация 6](#_Toc197449195)

[2.1 Dockerfile 6](#_Toc197449196)

[2.2 docker-compose 7](#_Toc197449197)

[Заключение 10](#_Toc197449198)

[Список использованных источников 11](#_Toc197449199)

[Приложение А 12](#_Toc197449200)

[Приложение Б 14](#_Toc197449201)

[Приложение В 15](#_Toc197449202)

[Приложение Г 17](#_Toc197449203)

# 

# ВВЕДЕНИЕ

В современном обществе наблюдается значительное увеличение ассортимента косметических средств и средств для ухода за собой. На рынке косметической продукции складывается острая конкуренция между производителями, продавцами и поставщиками, каждый из которых стремится привлечь внимание потребителей и добиться успеха за счет эффективных рекламных кампаний. Однако зачастую доверие к продуктам крупных брендов основывается не только на их реальной эффективности и безопасности, но и на популярности самой марки, что приводит к определенным перекосам в восприятии потребителей.

Одновременно с этим растут и требования потребителей. Современные пользователи ожидают от косметических средств не только соответствия высоким стандартам безопасности и использования натуральных компонентов, но и мгновенного видимого эффекта. Важным критерием также остается соотношение цены и качества, что делает выбор подходящего продукта еще более сложной задачей.

С развитием цифровых технологий многие аспекты повседневной жизни стали более простыми и удобными. Значительный вклад в это внесло появление индивидуальных помощников и систем автоматизации, способных анализировать данные и предлагать оптимальные решения на основании заданных пользователем параметров.

**Целью** практики являлась реализация серверной части индивидуального помощника, который сможет выполнить роль виртуального консультанта при подборе косметических средств для ухода за собой. Серверная часть была спроектирована и частично реализована в рамках предыдущей практики, поэтому в этом семестре работа фокусировалась вокруг ещё не реализованного алгоритма подбора косметических средств.

Для достижения цели были поставлены следующие **задачи**:

1. планирование процесса реализации,
2. реализация программного модуля, который включает в себя алгоритм подбора,
3. интеграция реализованного модуля в существующую систему,
4. упаковка системы в Docker («контейнеризация»).

Также дополнительными задачами на практику являлись:

1. выступление с докладом на конференции «Международная научная студенческая конференция — 2025» (приложение А),
2. написание текста ВКР.

Актуальность темы практики обусловлена тем, что у многих людей часто возникает потребность в выборе средств по уходу за собой, однако существует проблема, которая состоит в том, что не все разбираются в косметических средствах настолько хорошо, чтобы подобрать себе правильную линейку средств и оценить их состав при покупке и подборе. Это приводит к неэффективным покупкам, расходам времени, а также может повлечь применение опасных для здоровья косметических средств ввиду индивидуальных особенностей организма, таких как аллергия на определенные компоненты в составе косметических средств, например. Реализация серверной части системы для анализа и индивидуального подбора косметических средств позволит использовать её вместе с клиентской частью для решения обозначенных проблем.

**Сведения о месте прохождения практики:** место прохождения практики – ФГБУН Институт математики им. С. Л. Соболева СО РАН, Лаборатория теории вычислимости и прикладной логики, 630090, Новосибирская обл., Новосибирск, пр. Академика Коптюга, 4. Главная цель Института состоит в создании фундаментальных знаний в развиваемых им научных направлениях, разработке научных основ современной техники и технологии без цели извлечения прибыли.

# ОСНОВНАЯ ЧАСТЬ

## 1. Модуль подбора косметических средств

Данный модуль задействуется, когда нужно подобрать косметическое средство. Он использует векторную базу данных ChromaDB [1], которая запускается в докере. Эта база данных поддерживает HNSW-индексацию для быстрого поиска ближайших соседей, а также поддерживает возможность использовать различные функции подсчёта векторного расстояния (*косинусное расстояние*, *Евклидово расстояние*, *скалярное произведение*); в алгоритме подбора используется *Евклидово расстояние*.

Информация о косметических средства берётся с сайта «Золотое яблоко» [2]. Был произведён парсинг нескольких тысяч косметических средств, для которых было посчитано векторное представление и которые были сохранены в базу данных. Для парсинга «Золотого яблока» использовалась библиотека Playwright [3]. Онабыла выбранf, так как данный сайт использует динамическую загрузку контента с помощью JavaScript, что делает невозможным использование обычных библиотек для парсинга, таких как BeautifulSoup [4] или lxml [5]. Playwright позволяет эмулировать работу браузера, загружать страницы и извлекать данные с элементов, которые рендерятся динамически.

### 1.1 Реализация алгоритма

Алгоритм подбора основан на векторном представлении косметических средств. Векторное представление косметического средства включает в себя следующие признаки:

* 31 признак, связанным с назначением средства (от пигментных пятен, тонизирование, увлажнение и т.д.),
* 161 признак, связанный с типом продукта (маски для тела, мазь, шампунь, пилинг для тела и т.д.),
* 9 типов кожи, для которой предназначено средство (сухая, атопичная, зрелая, обычная и т.д.),
* 12 типов волос, для которых предназначено средство (ломкие, жирные, вьющиеся, окрашенные и т.д.).
* 1 признак для гипоаллергенности (есть ли аллергия),
* 2 признака для пола (мужской или нет, женский или нет).

Итого есть 216 булевых признаков, из которых следующим образом составляется векторное представление.

На вход алгоритма подаются следующие параметры:

1. признаки, по которым строится векторное представление запроса,
2. предпочитаемые бренды (опционально),
3. ценовой диапазон (опционально).

Последовательность действий алгоритма такая:

1. из полученных на вход признаков строится векторное представление *v*;
2. происходит обращение к векторной базе данных с запросом по поиску *N*1 косметических средств, векторные представления которых наиболее близки к *v* – *отбор кандидатов*;
3. полученные от векторной базы данных косметические средства фильтруются по брендам и цене;
4. из итогового набора показывается не более *N*2 косметических средств.

### 1.2 Подбор настраиваемых параметров алгоритма

*N*1 и *N*2 – настраиваемые параметры. Параметр *N*2 фактически отвечает за отображение результатов работы алгоритма на клиентской части, а *N*1 напрямую влияет на алгоритм. Параметр *N*1 подбирался по следующему критерию: на втором этапе из векторной базы данных будет извлечено не «слишком мало», и не «слишком много» кандидатов.

В каком плане не «слишком мало»? Возможна ситуация, что все извлечённые из векторной базы данных кандидаты будут отсеяны на третьем этапе и пользователю вернётся пустой список косметических средств, хотя в базе данных есть средства, векторное представление которых близко к представлению *v*, а их бренд и цена удовлетворяют запросу, но поскольку в данной ситуации *N*1 имеет относительно маленькое значение, то они даже не были извлечены из базы данных.

В каком плане не «слишком много»? Конечно, можно на втором шаге извлекать абсолютно все косметические средства из базы данных. Однако временная сложность третьего этапа это *O*(*N*), где *N* – количество извлечённых косметических средств. Из этого следует, что чем больше средств извлекается из базы данных на втором этапе, тем медленнее будет проходить третий этап, поэтому есть желание извлекать как можно меньше средств на втором этапе.

Эмпирически было получено, что *N*1 ~ 1000 даёт оптимальный результат в смысле не «слишком мало» и не «слишком много».

## 2. Контейнеризация

### 2.1 Dockerfile

Поскольку серверная часть написана на языке программирования Python 3.12, использует систему poetry [6] для управления зависимостями, а также представляет собой REST API, то она была упакована в Docker-контейнер через файл для Docker (приложение Б).

Фактически в этом файле происходит следующее: все файлы из рабочей директории копируются в файловую систему контейнера, далее в контейнере с помощью poetry устанавливаются все зависимости, а после серверная часть запускается с помощью веб-сервера uvicorn, который представляет собой ASGI, то есть обеспечивает взаимодействие между программой на Python и непосредственно веб-сервером, встроенным в uvicorn.

CMD представляет собой команду, которая непосредственно исполнится при запуске контейнера. В этой команде используются переменные окружения *GLAMCHECK\_HOST*, *GLAMCHECK\_PORT* и *LOG\_LEVEL*.

Переменная *GLAMCHECK\_HOST* представляет собой IP-адрес сетевого интерфейса, на котором веб-сервер будет принимать входящие соединения от клиентов («GLAMCHECK» — это название всего проекта, состоящего из серверной и клиентской частей); эта переменная окружения так же поддерживает значение 0.0.0.0, что означает, что веб-сервер будет слушать входящие соединения на всех доступных сетевых интерфейсах.

Переменная *GLAMCHECK\_PORT* представляет собой порт, на котором веб-сервер будет принимать входящие клиентские соединения.

Переменная *LOG\_LEVEL* – это уровень логирования для логов непосредственно от uvicorn. Согласно официальной документации [8] в uvicorn поддерживаются следующие значения: *critical*, *error*, *warning*, *info*, *debug*, *trace*. По умолчанию uvicorn логирует все входящие запросы от клиентов в терминал с уровнем логирования info, соответственно, например, записи в лог с уровнем *debug* не будут отображаться на терминал. Чтобы это исправить, можно явно указать *LOG\_LEVEL* равным *debug*. *LOG\_LEVEL* можно указывать и в «другую сторону», чтобы не отображать в терминале записи в логах с уровнем *trace*, *debug*, *info* и *warning*, а лишь *error* и *critical*, например.

### 2.2 docker-compose

Серверная часть использует три различные базы данных: реляционная (PostgreSQL), документно-ориентированная (MongoDB) и векторная (ChromaDB). Поэтому для удобства запуска нужно было написать файл *docker-compose.yml*, который вместе с конфигурационным файлом позволил бы с помощью лишь одной команды в терминале запускать и серверную часть, и необходимые для неё базы данных (приложение В). В этом файле объявлено четыре сервиса, то есть четыре программы, которые запустятся при исполнении команды запуска:

* + **postgres**. Сервис представляет собой реляционную базу данных PostgreSQL, в которой хранятся данные пользователей. Конфигурация осуществляется стандартными переменными окружения: *POSTGRES\_DB* – это имя базы данных, которая автоматически создастся при запуске сервиса; *POSTGRES\_USER* – это имя пользователя базы данных; *POSTGRES\_PASSWORD* – это пароль пользователя базы данных.
  + **mongo**. Сервис представляет собой документно-ориентированную базу данных MongoDB, в которой хранятся компоненты косметических средств, использующиеся для анализа состава косметических средств. Конфигурация осуществляется стандартными переменными окружения: *MONGO\_USER* – это имя пользователя базы данных, *MONGO\_PASSWORD* – это пароль пользователя базы данных.
  + **chroma**. Сервис представляет собой векторную базу данных, в которой хранятся косметические средства вместе с их векторными представлениями. Поскольку по умолчанию данная вектора база данных не поддерживает аутентификацию и авторизацию при использовании её в Docker, было принято решение использовать её расширение для поддержки аутентификации и авторизации. Для этого были указаны значения для следующих переменных окружения:
    - *CHROMA\_SERVER\_AUTHN\_CREDENTIALS\_FILE*. Значение *server.htpasswd* говорит о том, что векторной базе данных нужно поддерживать аутентификацию и авторизацию через имя пользователя и хэш его пароля, указанные в файле *server.htpasswd*. Этот файл был создан с помощью следующей команды:

docker run --rm --entrypoint htpasswd httpd:2 -Bbn $user $password >> server.htpasswd

где $user – имя пользователя, а $password – пароль пользователя. Данная команда создаст файл *server.htpasswd*, в котором вместе с именем пользователя будет записан хэш его пароля. Для того, чтобы Docker-контейнеру был доступен данный файл, был добавлен том (volume).

* *CHROMA\_SERVER\_AUTHN\_PROVIDER*. Переданное значение говорит о том, что для аутентификации нужно использовать стандартный механизм.
  + **glamcheck**. Данный сервис представляет собой непосредственно серверную часть разрабатываемого помощника, то есть веб-сервер, поддерживающий REST API. Он использует описанный ранее Dockerfile, что указано через секцию build, поскольку Dockerfile должен находиться в той же директории, что и файл docker-compose.yml.

Все 3 сервиса баз данных имеют тома (volume), то есть их данные сбрасываются на дисковое пространство вне Docker. Благодаря этому содержимое баз данных остаётся неизменных при их перезапуске. Для успешного запуска файл *docker-compose.yml* необходимы определённые переменные окружения, большая часть которых уже упоминалась ранее. Для запуска можно использовать файл с расширением .env, в котором будут указаны названия переменных и их значения. Пример такого файла находится в приложении Г.

# ЗАКЛЮЧЕНИЕ

В результате прохождения практики были выполнены следующие задачи:

1. распланирован процесс реализации,
2. реализован программный модуль, который включает в себя алгоритм подбора,
3. произведена интеграция реализованного модуля в существующую систему,
4. система была упакована в Docker,
5. написан текст ВКР,
6. по теме практики было выступление с докладом на конференции «Международная научная студенческая конференция — 2025».

Таким образом, была реализована серверная часть системы для анализа и индивидуального подбора косметических средств, которая может быть использована для создания приложения, которое будет включать в себя как клиентскую, так и серверную части.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Introduction to Chroma Documentation. URL: https://docs.trychroma.com/docs/overview/introduction (дата обращения: 29.04.2025).
2. Goldapple. URL: https://goldapple.ru/ (дата обращения: 29.04.2025).
3. Playwright Documentation. URL: https://playwright.dev/ (дата обращения: 29.04.2025).
4. Beautiful Soup Documentation. URL: https://beautiful-soup.readthedocs.io/en/latest/ (дата обращения: 29.04.2025).
5. lxml Documentation. URL: https://lxml.de/ (дата обращения: 29.04.2025).
6. Poetry - Python Dependency Management. URL: https://python-poetry.org/ (дата обращения: 29.04.2025).
7. Docker Documentation. URL: https://www.docker.com/ (дата обращения: 29.04.2025).
8. Uvicorn Documentation. URL: https://www.uvicorn.org/#command-line-options (дата обращения: 29.04.2025).

# ПРИЛОЖЕНИЕ А

(обязательное)

**ТЕЗИСЫ НА КОНФЕРЕНЦИЮ «Международная научная студенческая конференция — 2024»**

УДК 004.42, 004.62, 004.8

**Серверная часть индивидуального помощника**

**для подбора косметических средств**

К.П. Кондренко

Новосибирский государственный университет

У многих людей возникает потребность в выборе средств по уходу за собой, однако не все разбираются в косметических средствах настолько хорошо, чтобы подобрать себе правильную линейку средств и оценить их состав при покупке и подборе. Это приводит к неэффективным покупкам, расходам времени, а также может повлечь применение опасных для здоровья косметических средств ввиду индивидуальных особенностей организма, например, таких как аллергия на определенные компоненты в составе косметических средств. Семантический подход в разработке системы подбора косметических средств позволяет обрабатывать неточные и неполные данные, учитывать индивидуальные предпочтения пользователей и их оценки товаров [1].

В связи с этим в НГУ начата разработка индивидуального помощника, который сможет выполнить роль консультанта при подборе средств по уходу за собой [2]. Целью данного проекта является созданиесерверного модуля индивидуального помощника, который позволяет подбирать косметические средства по индивидуальным особенностям, а также поддерживает анализ состава косметических средств, аутентификацию и авторизацию пользователей.

Приложение позволит пользователям экономить время при подборе косметических средств, а также исключит потенциальные ошибки пользователей, поскольку подбор будет происходить автоматически по указанным индивидуальным особенностям.

Автоматический подбор косметических средств будет происходить с помощью разработанного алгоритма, использующего векторные представления пользователей и косметических средств, а анализ средств использует существующий веб-сервис с компонентами [3].

1. Yakhyaeva G., Skokova V. Subjective Expert Evaluations in the Model-Theoretic Representation of Object Domain Knowledge \\ Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2021, 12948 LNAI, pp. 152–165.
2. Егоренко К. В., Кондренко К. П. Разработка индивидуального помощника для подбора косметических средств \\ Тезисы докладов Международной конференции. Новосибирск, 2024. С. 53.
3. Косметическая база данных COSMOBASE.RU, онлайн анализатор косметики. URL: https://cosmobase.ru (дата обращения: 16.02.2025).

Научный руководитель — канд. физ.-мат. наук, доц. Г.Э. Яхъяева

# ПРИЛОЖЕНИЕ Б

(обязательное)

**Dockerfile серверной части**

FROM python:3.12-slim

WORKDIR /app

COPY . /app

RUN pip install poetry

RUN poetry config virtualenvs.create false \

&& poetry install --no-interaction --no-ansi

CMD uvicorn glamcheck.app:app --host ${GLAMCHECK\_HOST} --port ${GLAMCHECK\_PORT} --log-level ${LOG\_LEVEL}

# ПРИЛОЖЕНИЕ В

(обязательное)

**Файл docker-compose.yml**

services:

postgres:

image: postgres

container\_name: glamcheck-postgresql

environment:

POSTGRES\_DB: ${POSTGRES\_DB}

POSTGRES\_USER: ${POSTGRES\_USER}

POSTGRES\_PASSWORD: ${POSTGRES\_PASSWORD}

volumes:

- ./postgres-data:/var/lib/postgresql/data

ports:

- "${POSTGRES\_PORT}:5432"

mongodb:

image: mongo

container\_name: glamcheck-mongo

environment:

MONGO\_INITDB\_ROOT\_USERNAME: ${MONGO\_USER}

MONGO\_INITDB\_ROOT\_PASSWORD: ${MONGO\_PASSWORD}

volumes:

- ./mongo-data:/data/db

ports:

- "${MONGO\_PORT}:27017"

chroma:

image: chromadb/chroma

container\_name: glamcheck-chroma

environment:

- CHROMA\_SERVER\_AUTHN\_CREDENTIALS\_FILE=server.htpasswd

- CHROMA\_SERVER\_AUTHN\_PROVIDER=chromadb.auth.basic\_authn.BasicAuthenticationServerProvider

volumes:

- ./chroma-data:/chroma/chroma

- ./server.htpasswd:/chroma/server.htpasswd

ports:

- "${CHROMA\_PORT}:8000"

glamcheck:

build:

context: .

container\_name: glamcheck-app

volumes:

- .:/app

depends\_on:

- postgres

- mongodb

- chroma

env\_file:

- .env

ports:

- ${GLAMCHECK\_PORT}:${GLAMCHECK\_PORT}

expose:

- "${GLAMCHECK\_PORT}"

# ПРИЛОЖЕНИЕ Г

(обязательное)

**Пример конфигурационного файла .env**

POSTGRES\_DB=glamcheck

POSTGRES\_USER=glamcheck

POSTGRES\_PASSWORD=postgresql-password

POSTGRES\_PORT=5432

MONGO\_PORT=27017

MONGO\_USER=glamcheck

MONGO\_PASSWORD=mongo-password

CHROMA\_PORT=8080

GLAMCHECK\_HOST=0.0.0.0

GLAMCHECK\_PORT=5555

LOG\_LEVEL=error