**АННОТАЦИЯ**

Данный программный документ содержит пояснительную записку к программе «CollaborativeRecommenderSystem» («Рекомендательная система для веб-приложений на основе метода коллаборативной фильтрации»), которая служит для просчёта и показа рекомендованных объектов на основе действий пользователей системы. В разделе «Введение» текущего документа указано наименование программы и документы, которые были взяты в основу при разработке. Раздел «Назначение и область применения» содержит функциональное и эксплуатационное назначение ПО, а также краткую характеристику его области применения. Раздел «Технические характеристики» описывает постановку задачи на разработку программы, алгоритм ее функционирования, обоснование выбора схемы алгоритма решения задачи, состав входных и выходных данных, состав технических и программных средств. В разделе «Ожидаемые технико-экономические показатели» отражены планируемая потребность и экономические превосходства разработки по сравнению с отечественными и зарубежными образцами или аналогами.

Настоящий документ разработан в соответствии с требованиями:

1) ГОСТ 19.101-77 Виды программ и программных документов [1];

2) ГОСТ 19.102-77 Стадии разработки [2];

3) ГОСТ 19.103-77 Обозначения программ и программных документов [3];

4) ГОСТ 19.104-78 Основные надписи [4];

5) ГОСТ 19.105-78 Общие требования к программным документам [5];

6) ГОСТ 19.106-78 Требования к программным документам, выполненным печатным способом [6];

7) ГОСТ 19.404-79 Пояснительная записка. Требования к содержанию и оформлению [7].

**СОДЕРЖАНИЕ**

[**1. Введение 3**](#_Toc418514176)

[**1.1. Наименование программы 3**](#_Toc418514177)

[**1.2 документы, на основании которых ведется разработка 3**](#_Toc418514178)

[**2. Назначение и область применения 4**](#_Toc418514179)

[**2.1. Назначение программы 4**](#_Toc418514180)

[**2.1.1. Функциональное назначение 4**](#_Toc418514181)

[**2.1.2. Эксплуатационное назначение 4**](#_Toc418514182)

[**3. Технические характеристики 5**](#_Toc418514183)

[**3.1. Постановка задачи на разработку программы 5**](#_Toc418514184)

[**3.2. Описание алгоритма и функционирования программы 5**](#_Toc418514185)

[**3.2.1. Описание алгоритма программы 5**](#_Toc418514186)

[**3.2.2. Обоснование выбора алгоритма решения задачи 8**](#_Toc418514187)

[**3.2.3. Возможные взаимодействия программы с другими программами 8**](#_Toc418514188)

[**3.3. Описание и обоснование выбора метода организации входных и выходных данных 8**](#_Toc418514189)

[**3.3.1. Описание метода организации входных и выходных данных 8**](#_Toc418514190)

[**3.3.2. Обоснование выбора метода организации входных и выходных данных 9**](#_Toc418514191)

[**3.4. Описание и обоснование выбора состава технических и программных средств 9**](#_Toc418514192)

[**3.4.1. Состав технических и программных средств 9**](#_Toc418514193)

[**4. Технико-экономические показатели 11**](#_Toc418514194)

[**4.1. Предполагаемая потребность 11**](#_Toc418514195)

[**4.2. Экономические преимущества разработки по сравнению с отечественными и зарубежными аналогами 11**](#_Toc418514196)

[**5. Источники, использованные при разработке 12**](#_Toc418514197)

**Приложение 1.** [**Описание и функциональное назначение классов 13**](#_Toc418514198)

**Приложение 2.** [**Описание и функциональное назначение методов, полей и свойств 16**](#_Toc418514199)

**Приложение 3.** [**Диаграмма классов 25**](#_Toc418514200)

**Приложение 4.** [**Диаграмма вариантов использования 30**](#_Toc418514201)

**Приложение 5.** [**Диаграмма деятельности 31**](#_Toc418514202)

**Приложение 6.** [**Диаграмма кооперации 32**](#_Toc418514203)

**Приложение 7.** [**Диаграмма состояний 33**](#_Toc418514204)

**Приложение 8.** [**Диаграмма последовательностей 34**](#_Toc418514205)

**Приложение 9.** [**Диаграмма компонентов 35**](#_Toc418514206)

**Приложение 10.** [**Схема базы данных 36**](#_Toc418514207)

# ВВЕДЕНИЕ

* 1. **Наименование программы**

Наименование программы – «Рекомендательная система для веб-приложений на основе метода коллаборативной фильтрации». Краткое именование – «CollaborativeRecommenderSystem».

## Документы, на основании которых ведется разработка

Приказ Национального исследовательского университета "Высшая школа экономики"   
№ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ от \_\_\_\_\_\_\_\_\_\_\_\_\_.ХХХХХ

# НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ

* 1. **Назначение программы**
     1. **Функциональное назначение**

Система обеспечивает поиск и показ материалов, которые могли бы быть интересны текущему пользователю. Это происходит за счет выявления пользователей с похожими предпочтениями и анализом контента, что в совокупности дает релевантные результаты.

* + 1. **Эксплуатационное назначение**

Использование рекомендательных систем в веб-приложениях способствует увеличению заинтересованности пользователя в контенте сайта. Кроме того, это означает улучшение показателя среднего времени пользования сайтом для посетителя. Такое преимущество может быть использовано владельцами сайтов для показа рекламы или предложения специальных акций, так как пользователи будут более предрасположенными к согласию на них.

# ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ

* 1. **Постановка задачи на разработку программы**

Задание на курсовую работу. Тема работы: «Рекомендательная система для веб-приложений на основе метода коллаборативной фильтрации». Цель разработки программы: предоставить конечному пользователю систему, которая может использоваться для просчета и показа рекомендаций на основе метода коллаборативной фильтрации.

* 1. **Описание алгоритма и функционирования программы**
     1. **Описание алгоритма программы**

Система представляет собой веб-приложение, запускаемое на сервере, взаимодействие с которым происходит при помощи веб-браузера. Общая схема работы системы приведена ниже (рис. 1):

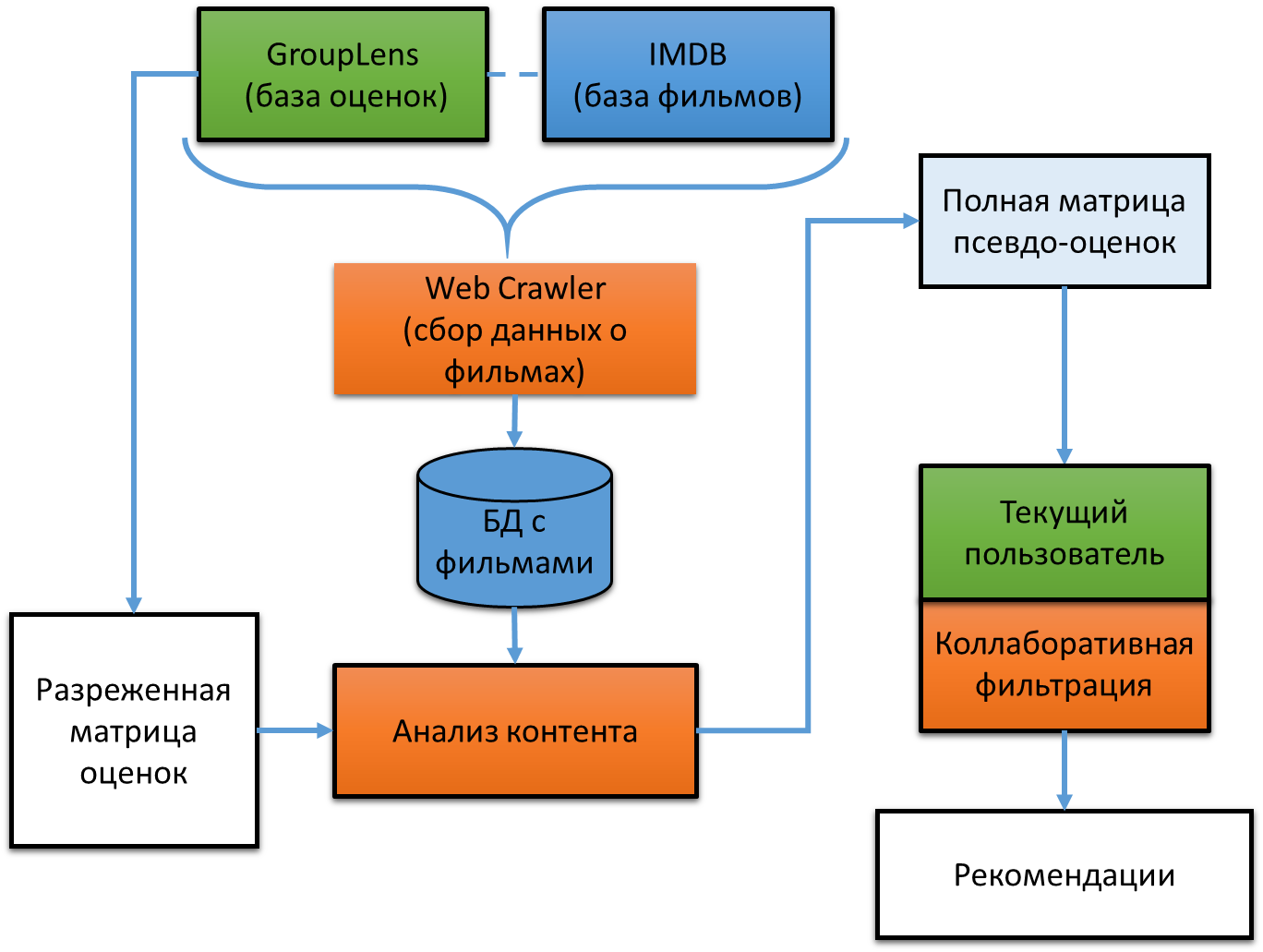


Рисунок 1

* + - 1. **Описание общей схемы работы системы**

1. Изначально система заполняется данными об оцененных пользователями фильмами по пятибалльной шкале. Набор данных предоставлен научно-исследовательской лабораторией GroupLens [8]. Гарантируется, что каждый пользователь оценил не менее двадцати фильмов.
2. Естественно, что каждый пользователь не мог оценить все фильмы, поэтому матрица оценок (пользователь/фильмы) содержит пропуски – матрица является разреженной.
3. Для решения этой проблемы используется система прогнозирования рейтингов на основе анализа контента фильмов. В качестве источника данных о фильмах используется портал IMDB [9]. Данные агрегируются и трансформируются в формат внутренней базы данных (приложение 10).
4. На этом этапе запускается анализ контента фильмов и высчитываются предсказания для незаполненных элементов матрицы оценок. Работа этого алгоритма описана в пункте 3.2.1.2. Так как эти оценки не являются априорными, будем называть их псевдо-оценками.
5. Далее выбирается пользователь (именуется как «текущий пользователь»), для которого необходимо просчитать рекомендации. При помощи алгоритма модифицированной коллаборативной фильтрации (пункт 3.2.1.3) прогнозируются оценки для фильмов, из которых отбираются N самых высоких. Эти фильмы и являются рекомендациями.
   * + 1. **Алгоритм предсказания оценок на основе анализа контента**

Каждый фильм в базе данных описан следующими текстовыми признаками: название, актеры, жанр, ключевые слова. Для того, чтобы оценить, какую оценку мог быть поставить пользователь фильму , можно посчитать, как похожи на фильм те, которые уже оценил пользователь , и сделать предположение о наиболее вероятной оценке. Эта идея лежит в основе байесовского классификатора.

Пусть – объект, представляющий фильм с признаками, описанными выше, а ­– класс объекта, в данном случае – оценка фильма (целое число от 1 до 5). Тогда по теореме Байеса вероятность того, что фильм принадлежит классу , равна , где – априорная вероятность принадлежать классу , – априорная вероятность объекта , ­– правдоподобие объекта на классе . Также считаем, что все признаки объекта независимы (наивный метод), то есть .

Перед обучением классификатора необходимо преобразовать каждый объект обучающей выборки в словарь, где ключами будут являться значения всех признаков всех объектов выборки, а значением – сколько раз ключ встречается в значениях признаков текущего объекта. Например:

На этапе обучения необходимо просчитать следующие величины:

1. для всех классов
2. для всех признаков и классов: (доля j-го признака среди всех признаков для объектов класса ). Используется (add-1) сглаживание на случай, если j‑ый признак не встречается ни разу.

Объект, для которого нужно предсказать класс, также нужно предварительно преобразовать: все значения всех признаков объекта становятся признаками этого объекта:

Задача наивного байесовского классификатора: , то есть определить класс тестового объекта, на котором классификатор возвращает наибольшую вероятность.

* + - 1. **Алгоритм модифицированной коллаборативной фильтрации**

1. Для начала нужно найти множество пользователей, которое максимально похоже на текущего по своим вкусам. Пользователи этого множества называются соседями текущего пользователя. Похожесть между пользователями и считается при помощи коэффициента корреляции Пирсона: , где – оценка фильма , поставленная пользователем , – общее количество фильмов. Затем данные величины ранжируются по убыванию и выбираются первые 30 пользователей [10], которые являются соседями текущего пользователя.
2. На этом шаге вводится понятие псевдо-вектора оценок. Назовем – псевдо-вектором оценок пользователя , если для всех фильмов выполняется следующее правило: , где – оценка фильму , которую поставил пользователь , – оценка, предсказанная на основе анализа контента. Будем использовать вектор псевдо-оценок для оценки похожести пользователей (см. предыдущий пункт) вместо реальных оценок.
3. Качество корреляции зависит от того, какие оценки использовались при подсчете и сколько их было. Для этого введем корректирующие коэффициенты:
   1. Смысловое взвешивание (Significance Weighting):   
      [10, 11]. Где – число фильмов, реально оцененных обоими пользователями.
   2. Среднегармоническое взвешивание (Harmonic Mean Weighting): , где [10,11], – число фильмов, реально оцененных пользователем . Смысл заключается в том, что чем меньше фильмов оценил пользователь, тем менее надежен вектор псевдо-оценок и, следовательно, коэффициент корреляции.

Общий коэффициент (гибридное взвешивание):

1. Также введем вес, с которым вектор псевдо-оценок текущего пользователя участвует в итоговой формуле просчёта рекомендаций (Self Weighting): , – число фильмов, реально оцененных пользователем , max = 2 в данном случае.
2. Итоговая формула предсказания оценки пользователем для фильма :

– число соседей, – коэффициент корреляции Пирсона, описанный выше. Знаменатель служит для нормализации суммы весов.

* + 1. **Обоснование выбора алгоритма решения задачи**

Алгоритм модифицированной коллаборативной фильтрации, несмотря на простоту реализации, дает хорошие результаты, так как совмещает в себе два метода: чистая коллаборативная фильтрация и предсказания на основе анализа контента.

Рассмотрим временную сложность второго метода:

для пользователя и фильма она составит , где – количество различных классов (в данном случае количество оценок, равно 5), ­– количество оцененных пользователем фильмов, – количество признаков у фильма .

Рассмотрим временную сложность первого метода:

Предварительно вычислим корреляции между всеми пользователями: , где – количество пользователей, – количество фильмов. Тогда весь метод будет иметь временную сложность , где neighbourhood – множество соседей пользователя.

Общая временная сложность для U пользователей и M фильмов (на практике требуется просчитывать предсказания лишь для еще не оцененных данным пользователем фильмов): .

Также стоит отметить, что алгоритм на основе анализа контента может быть легко распараллелен в силу предположения о независимости признаков.

* + 1. **Возможные взаимодействия программы с другими программами**

Взаимодействия с другими программами отсутствуют.

* 1. **Описание и обоснование выбора метода организации входных и выходных данных**
     1. **Описание метода организации входных и выходных данных**

Информация о фильмах должна быть предварительно загружена в базу данных системы. Информация об оценках так же может быть предварительно загружена в базу данных, либо внесена вручную при помощи интерфейса клиентской части. Схема базы данных описана в приложении 10. Некоторые этапы алгоритма просчёта рекомендаций кэшируются в базе данных для уменьшения времени его работы. Результаты работы алгоритма доступны для просмотра через интерфейс клиентской части системы, экспорт данных не предполагается.

* + 1. **Обоснование выбора метода организации входных и выходных данных**

В качестве источника данных служит реляционная SQL база данных, что способствует высокой скорости доступа к данным даже при одновременной работе с системой. Кроме того, наличие библиотек, позволяющих автоматически производить преобразование между объектами внутри системы и базой данной, делает работу с этим источником данных практичной и не затратной.

* 1. **Описание и обоснование выбора состава технических и программных средств**
     1. **Состав технических и программных средств**

Состав технических средств, необходимых для работы системы:

1. Серверная часть:
   1. Рекомендуется компьютер, оснащенный процессором с частотой 2 ГГц или более (двухъядерная система).
   2. Рекомендуется 2ГБ ОЗУ или более.
   3. Место на жестком диске: минимум 6 ГБ.
   4. Требуется монитор Super VGA с разрешением 800x600 пикселей или более высоким.
   5. Мышь Microsoft или совместимое указывающее устройство и клавиатура.
   6. CD/DVD-ROM привод.
   7. Доступ к интернету.
2. Клиентская часть:
   1. Устройство, поддерживающее выход в интернет.
   2. Рекомендуемая диагональ экрана: от семи дюймов и выше.
   3. Мышь или совместимое указывающее устройство и клавиатура.

Состав программных средств, необходимых для работы системы:

1. Серверная часть:
   1. Операционная система Microsoft Windows Server 2008/Vista/7/8/8.1, Mac OS X (10.8.3+, 10.9+), Linux (Ubuntu 12.04, Oracle Linux 5+, Red Hat Enterprise Linux 5.5+, 6.x).
   2. Сервер JavaEE приложений.
   3. Java 8;
   4. SQL-база данных MySQL;
2. Клиентская часть:
   1. Браузер, поддерживающий HTML5 спецификацию.
3. **ТЕХНИКО-ЭКОНОМИЧЕСКИЕ ПОКАЗАТЕЛИ**
   1. **Предполагаемая потребность**

Заинтересовать пользователя – задача владельца любого продукта. В виртуальном пространстве это можно сделать ненавязчиво и эффективно благодаря отслеживанию интересов пользователей. На этом методе и основана данная рекомендательная система. Независимость от вида рекомендуемых объектов делает ее привлекательной для владельцев посещаемых интернет-сайтов.

* 1. **Экономические преимущества разработки по сравнению с отечественными и зарубежными аналогами**

Несмотря на то, что рекомендательных систем существует большое количество, не все из них доступны для свободного скачивания. Данное приложение не требует денежных затрат на установку и использование и распространяется бесплатно.

1. **ИСТОЧНИКИ, ИСПОЛЬЗОВАННЫЕ ПРИ РАЗРАБОТКЕ**
2. ГОСТ 19.101-77 Виды программ и программных документов. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
3. ГОСТ 19.102-77 Стадии разработки. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
4. ГОСТ 19.103-77 Обозначения программ и программных документов. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001
5. ГОСТ 19.104-78 Основные надписи. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
6. ГОСТ 19.105-78 Общие требования к программным документам. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
7. ГОСТ 19.106-78 Требования к программным документам, выполненным печатным способом. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001
8. ГОСТ 19.404-79 Пояснительная записка. Требования к содержанию и оформлению. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
9. MovieLens | GroupLens [Электронный ресурс]// URL: <http://grouplens.com/datasets/movielens> (Дата обращения: 30.03.2015, режим доступа: свободный)
10. IMDB [Электронный ресурс]// URL: <http://www.imdb.com/> (Дата обращения: 30.03.2015, режим доступа: свободный)
11. Francesco Ricci and Lior Rokach and Bracha Shapira, Introduction to Recommender Systems Handbook, Recommender Systems Handbook (англ.) // Springer, 2011. – С. 1 – 35
12. Melville P.,Mooney R., Nagarajan R. Content-Boosted Collaborative Filtering for Improved Recommendations (англ.) // University of Texas, USA : Материалы конф. / AAAI-02, Austin, TX, USA, 2002. — 2002. — С. 187-192.
13. Yehuda Koren Factor in the Neighbors: Scalable and Accurate Collaborative Filtering (англ.) // Yahoo! Research, Haifa : журнал. — 2009. — С. 1 – 11.
14. Improvement of Collaborative Filtering with the Simple Bayesian Classifier [Электронный ресурс]// URL: [www.ics.uci.edu/~pazzani/Publications/IPSJ.pdf](http://www.ics.uci.edu/~pazzani/Publications/IPSJ.pdf) (Дата обращения: 01.05.2015, режим доступа: свободный)
15. Performance of Recommender Algorithms on Top-N Recommendation Tasks [Электронный ресурс]// URL: <http://labs.yahoo.com/publication/performance-of-recommender-algorithms-on-top-n-recommendation-tasks/> (Дата обращения: 01.05.2015, режим доступа: свободный)

|  |  |
| --- | --- |
| **ОПИСАНИЕ И ФУНКЦИОНАЛЬНОЕ НАЗНАЧЕНИЕ КЛАССОВ** | **ПРИЛОЖЕНИЕ 1** |

Таблица 1 – Классы пакета edu.kulikov.bayes

|  |  |
| --- | --- |
| **Класс** | **Назначение** |
| *BayesDatasetCreator* | Создание тренировочных и тестовых наборов данных, и запись их на диск |

Таблица 2 – Классы пакета edu.kulikov.collaborative

|  |  |
| --- | --- |
| **Класс** | **Назначение** |
| *CollaborativeParams* | Агрегирование параметров коллаборативной фильтрации, описанных в файле конфигурации |

Таблица 3 – Классы пакета edu.kulikov.config

|  |  |
| --- | --- |
| **Класс** | **Назначение** |
| *ApplicationConfig* | Задает путь до файла конфигурации настроек приложения |
| *BayesConfig* | Читает параметры байесовского классификатора из файла конфигурации настроек приложения и создает класс *BayesDatasetCreator* |
| *CollaborativeConfig* | Читает параметры коллаборативной фильтрации из файла конфигурации настроек приложения и создает класс *CollaborativeParams* |
| *JpaConfig* | Читает параметры работы c базой данных из файла конфигурации настроек приложения и применяет их |
| *MyAsyncConfigurer* | Задает настройки создания пула потоков |
| *SecurityConfig* | Задает настройки доступа к модулям приложения разным группам пользователей |
| *WebAppInitializer* | Задает параметры инициализации приложения и указывает конфигурационные классы |
| *WebMvcConfig* | Задает параметры клиентской части приложения |

Таблица 4 – Классы пакета edu.kulikov.controllers

|  |  |
| --- | --- |
| **Класс** | **Назначение** |
| *HomeController* | Обработчик домашней страницы приложения |
| *MovieController* | Обработчик запросов информации о фильмах |
| *PredictionController* | Обработчик запросов, связанных с просчетом рекомендаций |
| *RatingController* | Обработчик запросов, связанных с информацией о рейтингах |
| *SigninController* | Обработчик запросов, связанных с авторизацией пользователей |

Продолжение таблицы 4

|  |  |
| --- | --- |
| **Класс** | **Назначение** |
| *SignupController* | Обработчик запросов, связанных с регистрацией пользователей |
| *UserController* | Обработчик запросов, связанных c получением информации о пользователях |
| *UserCorrelationController* | Обработчик запросов, связанных с просчётом и получением корреляций между пользователями |
| *MetricsController* | Обработчик запросов, связанных с тестированием качества классификации |

Таблица 5 – Классы пакета edu.kulikov.dao

|  |  |
| --- | --- |
| **Класс** | **Назначение** |
| *AccountRepository* | Поддержка информации об учетных записях в базе данных |
| *MovieRepository* | Поддержка информации о фильмах в базе данных |
| *PredictionRepository* | Поддержка информации о предсказаниях оценок в базе данных |
| *RatingRepository* | Поддержка информации об оценках фильмов в базе данных |
| *UserCorrelationRepository* | Поддержка информации о корреляции фильмов в базе данных |
| *UserRepository* | Поддержка информации о пользователях в базе данных |
| *MetricRepository* | Поддержка информации о тестах в базе данных |

Таблица 6 – Классы пакета edu.kulikov.models

|  |  |
| --- | --- |
| **Класс** | **Назначение** |
| *Account* | Модель, описывающая учетную запись пользователя |
| *Actor* | Модель, описывающая актера фильма |
| *Genre* | Модель, описывающая жанр фильма |
| *Keyword* | Модель, описывающая ключевое слово фильма |
| *Movie* | Модель, описывающая фильм |
| *Metric* | Модель, описывающая результаты теста |
| *Prediction* | Модель, описывающая предсказание оценки для пары «пользователь» - «фильм» |
| *Rating* | Модель, описывающая оценку для пары «пользователь» - «фильм» |
| *User* | Модель, описывающая пользователя |
| *UserCorrelation* | Модель, описывающая корреляцию между двумя пользователями |

Таблица 7 – Классы пакета edu.kulikov.services

|  |  |
| --- | --- |
| **Класс** | **Назначение** |
| *AccountService* | Бизнес-логика работы с учетными записями |
| *BayesClassifierService* | Бизнес-логика работы с баейсовским классификатором |
| *BoostedPredictionServiceExecutor* | Контроль вычисления предсказаний при помощи модифицированной коллаборативной фильтрации |
| *ContentPredictionServiceExecutor* | Контроль вычисления предсказаний при помощи метода на основе анализа контента |
| *MovieService* | Бизнес-логика работы с фильмами |
| *PredictionService* | Бизнес-логика работы с информацией о предсказаниях оценок |
| *RatingService* | Бизнес-логика работы с информацией об оценках |
| *UserCorrelationService* | Бизнес-логика работы с информацией о корреляциях между пользователями |
| *UserCorrelationServiceExecutor* | Контроль вычисления корреляций между пользователями |
| *UserService* | Бизнес-логика работы с пользователями |
| *MetricService* | Бизнес-логика работы с информацией о тестах |
| *MetricServiceExecutor* | Создание тестового окружения и контроль выполнения очереди тестов |

Таблица 8 – Классы пакета edu.kulikov.view\_model

|  |  |
| --- | --- |
| **Класс** | **Назначение** |
| *BoostedPredictionProgress* | Модель, описывающая прогресс вычисления предсказаний при помощи модифицированной коллаборативной фильтрации |
| *ContentPredictionProgress* | Модель, описывающая прогресс вычисления предсказаний при помощи метода на основе анализа контента |
| *DonutPoint* | Модель, описывающая точку на круговой диаграмме |
| *MovieTitleStars* | Модель, описывающая строчку таблицы с оценками фильмов |
| *Page* | Модель, описывающая страницу при многостраничном отображении информации |
| *RatingPoint* | Модель, описывающая элемент столбчатой диаграммы |
| *SignupForm* | Модель, описывающая форму регистрации |
| *UserWithRatings* | Модель, описывающая пользователя и его общую статистику по оценкам (количество и средний индекс) |

|  |  |
| --- | --- |
| **ОПИСАНИЕ И ФУНКЦИОНАЛЬНОЕ НАЗНАЧЕНИЕ МЕТОДОВ, ПОЛЕЙ И СВОЙСТВ** | **ПРИЛОЖЕНИЕ 2** |

Таблица 9 – Описание членов класса *BayesDatasetCreator*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Методы** | | | | |
| *Имя* | *Мод. Доступа* | *Тип* | *Аргументы* | *Назначение* |
| exportRatedMovies | public | File | User, recreate | Экспорт оцененных фильмов пользователя в формат обучающей выборки |
| exportUnratedMovie | public | File | Movie | Экспорт фильма в формат тестируемой выборки |

Таблица 10 – Описание членов класса MovieController

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Методы** | | | | |
| *Имя* | *Мод. Доступа* | *Тип* | *Аргументы* | *Назначение* |
| getAll | public | ModelAndView | Integer | Получение списка фильмов для заданной страницы из каталога фильмов |
| getById | public | ModelAndView | Integer, Principal | Получение информации о фильме по его номеру |

Таблица 11 – Описание членов класса PredictionController

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Методы** | | | | |
| *Имя* | *Мод. Доступа* | *Тип* | *Аргументы* | *Назначение* |
| index | public | ModelAndView | Integer, Integer | Главная страница раздела для просчета рекомендаций |
| firstStep | public | ModelAndView | Integer, Integer | Показ страницы с первой стадией просчёта рекомендаций |
| secondStep | public | ModelAndView | Integer, Integer | Показ страницы со второй стадией просчёта рекомендаций |
| thirdStep | public | ModelAndView | Integer, Integer | Показ страницы с третьей стадией просчёта рекомендаций |
| contentPredictionTrigger | public | void |  | Запуск алгоритма предсказания оценок на основе анализа контента |
| stopContentPrediction | public | void |  | Остановка алгоритма предсказания оценок на основе анализа контента |
| getContentPredictionProgress | public | ContentPredictionProgress |  | Получение прогресса вычисления предсказаний оценок на основе анализа контента |
| correlationTrigger | public | void | Integer | Запуск алгоритма просчета корреляций для заданного пользователя |
| getCorrelationProgress | public | Integer | Integer | Получение прогресса вычисления корреляций для заданного пользователя |
| correlationStop | public | void | Integer | Остановка алгоритма просчета корреляций для заданного пользователя |
| boostedPredictionTrigger | public | void | Integer | Запуск алгоритма просчета рекомендаций на основе модифицированного метода коллаборативной фильтрации для заданного пользователя |
| getBoostedPredictionProgress | public | Integer | Integer | Получение прогресса вычисления рекомендаций на основе модифицированного метода коллаборативной фильтрации для заданного пользователя |
| stopBoostedPrediction | public | void | Integer | Остановка алгоритма просчета рекомендаций на основе модифицированного метода коллаборативной фильтрации для заданного пользователя |

Таблица 12 – Описание членов класса RatingController

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Методы** | | | | |
| *Имя* | *Мод. Доступа* | *Тип* | *Аргументы* | *Назначение* |
| index | public | ModelAndView |  | Получение первой страницы со статистикой по оценкам |
| getByStars | public | ModelAndView | Integer | Получение страницы со статистикой для заданной оценки (от 1 до 5) |
| saveRating | public | void | Integer, Integer, HttpServletRequest, Principal | Сохранение информации об оценке в базе данных |

Таблица 12 – Описание членов класса UserController

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Методы** | | | | |
| *Имя* | *Мод. Доступа* | *Тип* | *Аргументы* | *Назначение* |
| getAll | public | ModelAndView | Integer | Получение списка пользователей для заданной страницы из каталога пользователей |
| getById | public | ModelAndView | Integer | Получение информации о пользователе по его номеру |
| getById | public | ModelAndView | String | Получение информации о пользователе по логину его учетной записи |

Таблица 13 – Описание членов класса UserCorrelationController

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Методы** | | | | |
| *Имя* | *Мод. Доступа* | *Тип* | *Аргументы* | *Назначение* |
| fillUserCorrelations | public | double | Integer, Integer | Просчет и сохранение корреляции для двух заданных пользователей |
| getTopCorrelations | public | List<UserCorrelation> | Integer | Получение top N пользователей с максимальной корреляцией с заданным пользователем |

Таблица 14 – Описание членов класса AccountRepository

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Методы** | | | | |
| *Имя* | *Мод. Доступа* | *Тип* | *Аргументы* | *Назначение* |
| save | public | Account | Account | Сохранение учетной записи в базе данных |
| findByLogin | public | Account | String | Поиск учетной записи по логину |
| initialize | public | void | Account, EnumSet<Account.LAZY> | Заполнение полей с отложенной инцициализацией |

Таблица 15 – Описание членов класса MovieRepository

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Методы** | | | | |
| *Имя* | *Мод. Доступа* | *Тип* | *Аргументы* | *Назначение* |
| getById | public | Movie | id, EnumSet<Movie.LAZY> | Получение фильма по его номеру |
| getAll | public | List<Movie> | int, int, EnumSet<Movie.LAZY> | Получение определенного числа фильмов начиная с заданного индекса |
| getUnratedMovies | public | List<Movie> | User | Получение всех фильмов, которые не оценил пользователь |
| getCount | public | Long |  | Получение количества всех фильмов |

Таблица 16 – Описание членов класса PredictionRepository

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Методы** | | | | |
| *Имя* | *Мод. Доступа* | *Тип* | *Аргументы* | *Назначение* |
| getAverageContentPrediction | public | Double | User | Получение средней оценки для предсказанных при помощи метода на основе анализе контента для заданного пользователя |
| getPseudoStars | public | Integer | User, Movie | Получение псевдо-оценки для заданного пользователя и фильма |
| getAveragePseudoStarsForUser | public | Double | User | Получение среднего значения для вектора псевдо-оценок для заданного пользователя |
| getPredictionUserMovie | public | Prediction | User, Movie | Получение предсказания оценки для заданного пользователя и фильма |
| setContentPredictionOutdatedForUser | public | void | User | Помечает, что для текущего пользователя требуется пересчёт предсказаний на основе анализа контента |
| add | public | Prediction | Prediction | Сохранение предсказания в базе данных |
| delete | public | void | Prediction | Удаление предсказания из базы данных |
| getUpdatedPredictionsForUser | public | List<Prediction> | User | Получение предсказаний, для которых не требуется пересчет. |

Таблица 17 – Описание членов класса RatingRepository

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Методы** | | | | |
| *Имя* | *Мод. Доступа* | *Тип* | *Аргументы* | *Назначение* |
| getCommonMoviesRatings | public | Pair<List<Rating>, List<Rating>> | User, User | Получение оценок для фильмов, оцененных обоими пользователями |
| getAllOccupationsShareByStars | public | List<Pair<String, Integer>> | int | Получение статистики по оценкам, проставленным пользователями разных видов профессий |
| getAllAgesShareByStars | public | List<Pair<String, Integer>> | int | Получение статистики по оценкам, проставленным пользователями разных диапазонов возрастов |
| getCommonRatedMoviesCount | public | int | User, User | Получение количества фильмов, оцененных обоими пользователями |
| getAverageRating | public | Double | User | Средняя оценка, поставленная пользователем |
| getUserRatedCount | public | Long | User | Количество фильмов, оцененных пользователем |
| getRating | public | Rating | User, Movie | Получение оценки для пары «пользователь» - «фильм» |
| add | public | Rating | Rating | Добавление оценки в базу данных |
| update | public | Rating | Rating | Обновление оценки в базе данных |

Таблица 18 – Описание членов класса UserCorrelationRepository

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Методы** | | | | |
| *Имя* | *Мод. Доступа* | *Тип* | *Аргументы* | *Назначение* |
| getByTwoUsers | public | UserCorrelation | User, User | Получение корреляции для двух пользователей |
| getAllByUser | public | List<UserCorrelation> | User | Получение всех корреляций данного пользователя |
| getTopNByUser | public | List<UserCorrelation> | user, int | Получение top N корреляций для данного пользователя |
| add | public | UserCorrelation | UserCorrelation | Добавление корреляции в базу данных |
| update | public | UserCorrelation | UserCorrelation | Обновление корреляции в базе данных |
| setCorrelationsOutdated | public | void | User | Помечает корреляции для текущего пользователя как требующие пересчета |
| getOutdatedCorrelationsByUser | public | List<UserCorrelation> | User | Получение корреляций для пользователя, требующих персчёта |

Таблица 19 – Описание членов класса Account

|  |  |  |  |
| --- | --- | --- | --- |
| **Свойства** | | | |
| *Имя* | *Мод. Доступа* | *Тип* | *Назначение* |
| id | private | Long | Идентификатор учетной записи |
| login | private | String | Логин учетной записи |
| password | private | String | Пароль учетной записи |
| role | private | String | Роль учетной записи |
| user | private | User | Пользователь, привязанный к учетной записи |

Таблица 20 – Описание членов класса Actor

|  |  |  |  |
| --- | --- | --- | --- |
| **Свойства** | | | |
| *Имя* | *Мод. Доступа* | *Тип* | *Назначение* |
| id | private | Long | Идентификатор актера |
| name | private | String | Фамилия и имя актера |

Таблица 21 – Описание членов класса Genre

|  |  |  |  |
| --- | --- | --- | --- |
| **Свойства** | | | |
| *Имя* | *Мод. Доступа* | *Тип* | *Назначение* |
| id | private | Long | Идентификатор жанра |
| genre | private | String | Название жанра |

Таблица 22 – Описание членов класса Keyword

|  |  |  |  |
| --- | --- | --- | --- |
| **Свойства** | | | |
| *Имя* | *Мод. Доступа* | *Тип* | *Назначение* |
| id | private | Long | Идентификатор ключевого слова |
| word | private | String | Ключевое слово |

Таблица 23 – Описание членов класса Movie

|  |  |  |  |
| --- | --- | --- | --- |
| **Свойства** | | | |
| *Имя* | *Мод. Доступа* | *Тип* | *Назначение* |
| id | private | Long | Идентификатор фильма |
| title | private | String | Название фильма |
| year | private | int | Год выхода фильма |
| actors | private | List<Actor> | Список актеров фильма |
| genres | private | List<Genre> | Список жанров фильма |
| keywords | private | List<Keyword> | Список ключевых слов фильма |
| plot | private | String | Сюжет фильма |
| posterUrl | private | String | Ссылка на постер к фильму |
| imdbId | private | String | Номер фильма в базе IMDB |
| ratings | private | List<Rating> | Список оценок данного фильма |

Таблица 24– Описание членов класса Metric

|  |  |  |  |
| --- | --- | --- | --- |
| **Свойства** | | | |
| *Имя* | *Мод. Доступа* | *Тип* | *Назначение* |
| id | private | Long | Идентификатор теста |
| user | private | User | Пользователь, для которого проводилось тестирование |
| testSetCount | private | Integer | Размер тестовой выборки |
| trainSetCount | private | Integer | Размер тренировочной выборки |
| tpr | private | Double | Доля истинно положительных объектов |
| fpr | private | Double | Доля ложно положительных объектов |
| plot | private | String | Сюжет фильма |

Таблица 25 – Описание членов класса Prediction

|  |  |  |  |
| --- | --- | --- | --- |
| **Свойства** | | | |
| *Имя* | *Мод. Доступа* | *Тип* | *Назначение* |
| user | private | User | Пользователь, для которого делается прогноз оценки |
| movie | private | Movie | Фильм, для которого делается прогноз оценки |
| contentPrediction | private | Integer | Предсказание оценки на основе анализа контента |
| boostedPrediction | private | Integer | Предсказание оценки на основе модифицированной коллаборативной фильтрации |
| isContentUpdated | private | boolean | Значение истинно, если не требуется пересчет оценки на основе анализа контента |

Таблица 26 – Описание членов класса User

|  |  |  |  |
| --- | --- | --- | --- |
| **Свойства** | | | |
| *Имя* | *Мод. Доступа* | *Тип* | *Назначение* |
| id | private | int | Идентификатор пользователя |
| age | private | int | Фильм, для которого делается прогноз оценки |
| isMale | private | boolean | Предсказание оценки на основе анализа контента |
| occupation | private | String | Предсказание оценки на основе модифицированной коллаборативной фильтрации |
| contentPredictionUpdated | private | boolean | Значение истинно, если не требуется пересчет оценки на основе анализа контента |
| ratings | private | List<Rating> | Оценки, проставленные данным пользователем |

Таблица 27 – Описание членов класса BayesClassifierService

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Методы** | | | | |
| *Имя* | *Мод. Доступа* | *Тип* | *Аргументы* | *Назначение* |
| loadTrainingSet | public | Instances | int, boolean | Получение тренировочной выборки для заданного пользователя |
| loadTestSet | public | Instances | int | Получение тестовой выборки на основе данного фильма |
| trainClassifier | public | void | Instances | Обучение классификатора |
| getClassesPrediction | public | HashMap<String, Double> | Instances | Получение вероятностей распределения классов по тестовой выборке |
| setUpBayesClassifier | private | void |  | Установка параметров классификатора |

Таблица 28 – Описание членов класса BoostedPredictionServiceExecutor

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Методы** | | | | |
| *Имя* | *Мод. Доступа* | *Тип* | *Аргументы* | *Назначение* |
| doBoostedPrediction | public | void | int | Запуск алгоритма просчета рекомендаций на основе модифицированного метода коллаборативной фильтрации для заданного пользователя |
| getProgress | public | Double | int | Прогресс вычисления рекомендаций на основе модифицированного метода коллаборативной фильтрации для заданного пользователя |
| stop | public | void | int | Остановка алгоритма просчета рекомендаций на основе модифицированного метода коллаборативной фильтрации для заданного пользователя |

Таблица 29 – Описание членов класса ContentPredictionServiceExecutor

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Методы** | | | | |
| *Имя* | *Мод. Доступа* | *Тип* | *Аргументы* | *Назначение* |
| doContentPrediction | public | void |  | Запуск алгоритма просчета рекомендаций на основе анализа контента |
| getProgress | public | Double |  | Прогресс вычисления рекомендаций на основе анализа контента |
| getCurrentMovieId | public | Integer |  | Получение номера фильма, для которого производится просчет рекомендаций |
| getCurrentUserId | public | Integer |  | Получение номера пользователя, для которого производится просчет рекомендаций |

Таблица 30 – Описание членов класса UserCorrelationServiceExecutor

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Методы** | | | | |
| *Имя* | *Мод. Доступа* | *Тип* | *Аргументы* | *Назначение* |
| doUserCorrelation | public | void | int | Запуск алгоритма просчета корреляций для данного пользователя |
| getProgress | public | Double | int | Прогресс вычисления корреляций для данного пользователя |
| stop | public | void | int | Остановка алгоритма просчета корреляций для данного пользователя |

Таблица 31 – Описание членов класса MetricService

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Методы** | | | | |
| *Имя* | *Мод. Доступа* | *Тип* | *Аргументы* | *Назначение* |
| getRoc4 | public | Pair<Double, Double> | Integer, Integer | Просчет значений True Positive Rate и False Positive Rate для тестового пользователя, используя пороговое значение 4 |
| getById | public | Metric | Integer | Получение теста по его идентификатору |
| getAll | public | List<Metric> |  | Получение всех тестов |
| save | public | Metric | Metric | Сохранение модели теста в базе данных |

|  |  |
| --- | --- |
| **ДИАГРАММА КЛАССОВ** | **ПРИЛОЖЕНИЕ 3** |

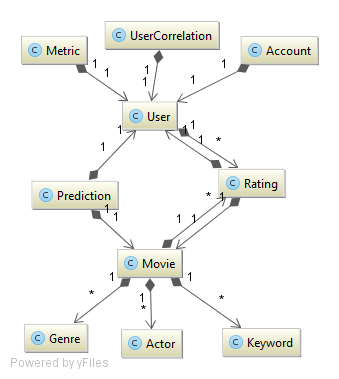


Рисунок 2 – Диаграмма модельных классов

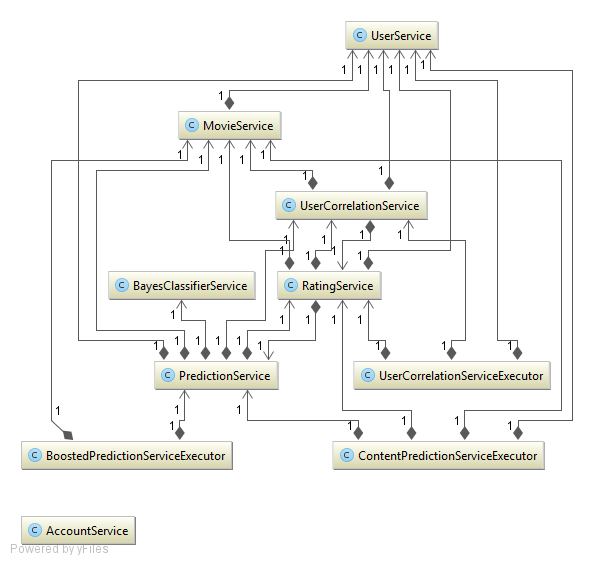


Рисунок 3 – Диаграмма сервисных классов

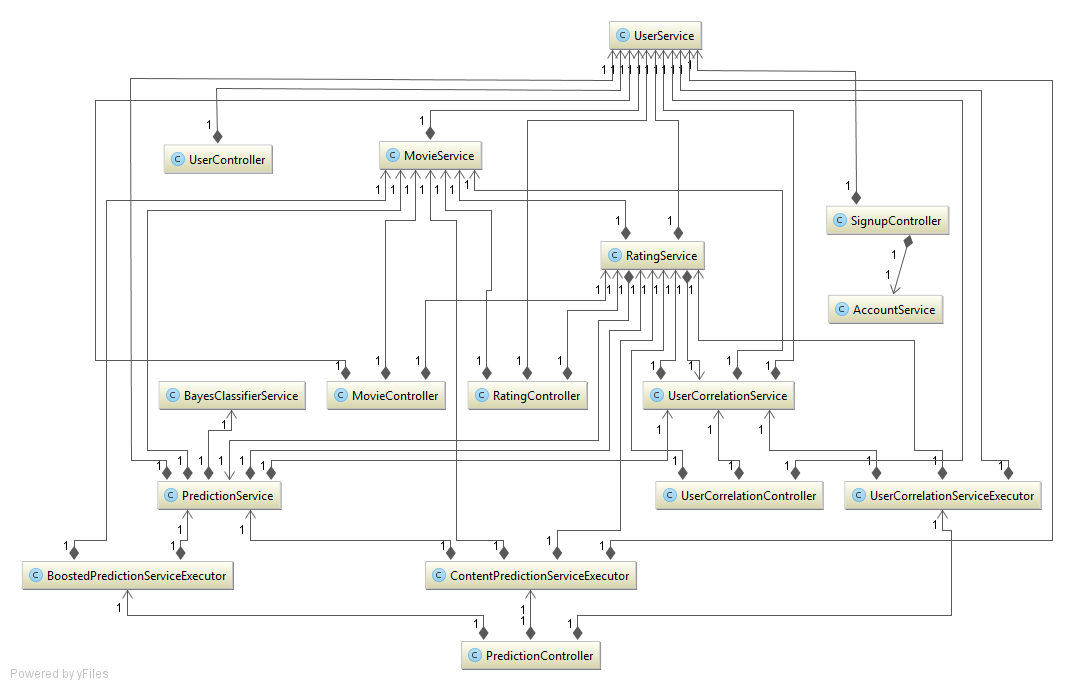


Рисунок 4 – Диаграмма контроллеров и сервисных классов

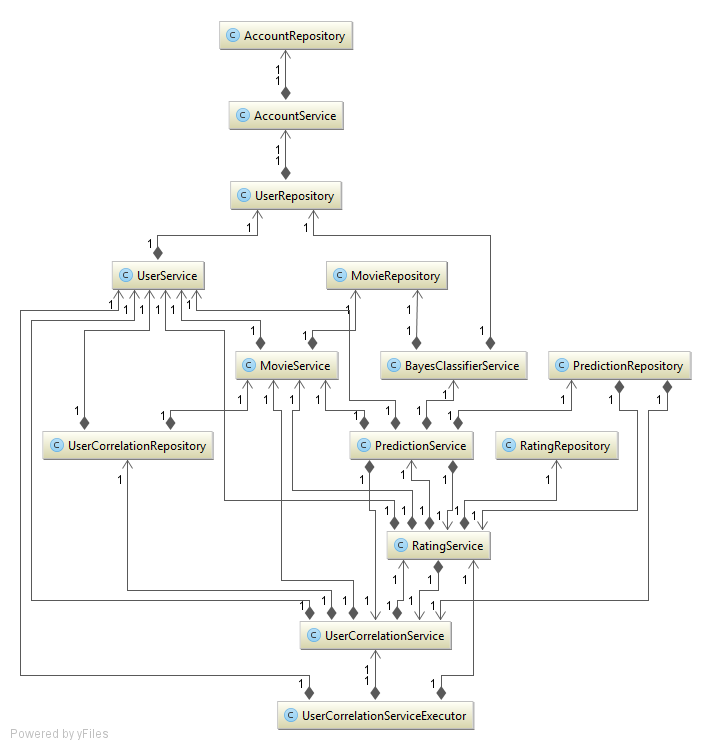


Рисунок 5 – Диаграмма репозиториев и сервисных классов

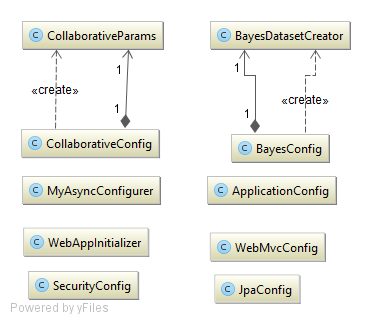


Рисунок 6 – Диаграмма классов конфигураций

|  |  |
| --- | --- |
| **ДИАГРАММА ВАРИАНТОВ ИСПОЛЬЗОВАНИЯ** | **ПРИЛОЖЕНИЕ 4** |

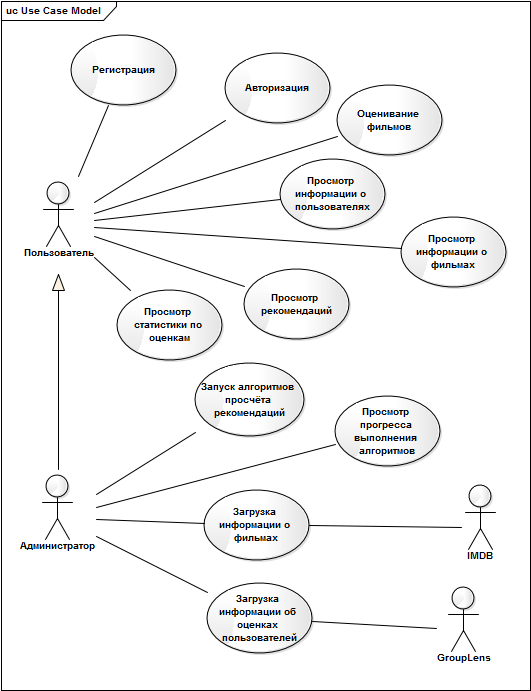


Рисунок 7 – Диаграмма вариантов использования

|  |  |
| --- | --- |
| **ДИАГРАММА ДЕЯТЕЛЬНОСТИ** | **ПРИЛОЖЕНИЕ 5** |

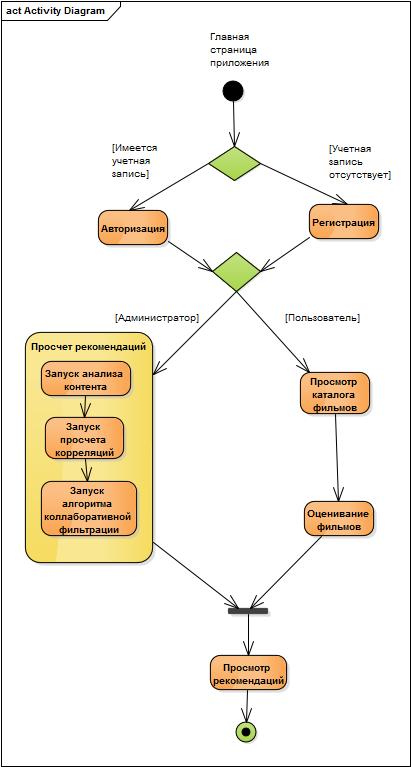


Рисунок 8 ­– Диаграмма деятельности

|  |  |
| --- | --- |
| **ДИАГРАММА КООПЕРАЦИИ** | **ПРИЛОЖЕНИЕ 6** |

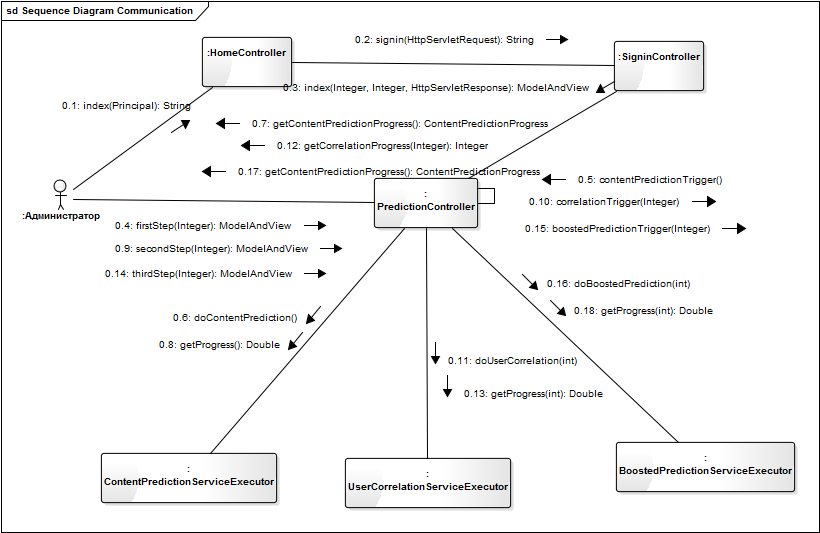


Рисунок 9 ­– Диаграмма кооперации

|  |  |
| --- | --- |
| **ДИАГРАММА СОСТОЯНИЙ** | **ПРИЛОЖЕНИЕ 7** |

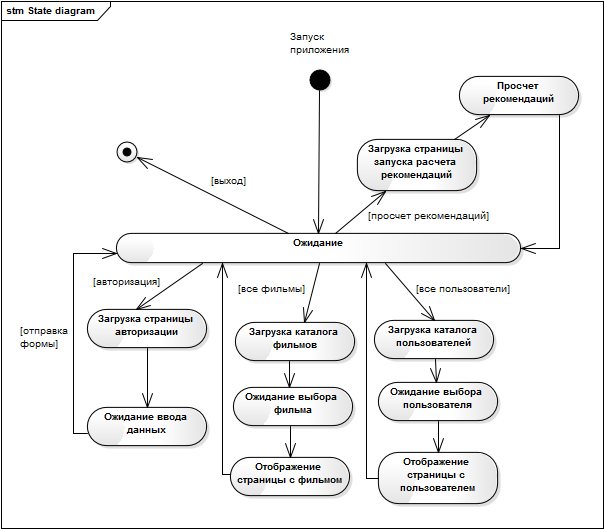


Рисунок 10 – Диаграмма состояний

|  |  |
| --- | --- |
| **ДИАГРАММА ПОСЛЕДОВАТЕЛЬНОСТЕЙ** | **ПРИЛОЖЕНИЕ 8** |

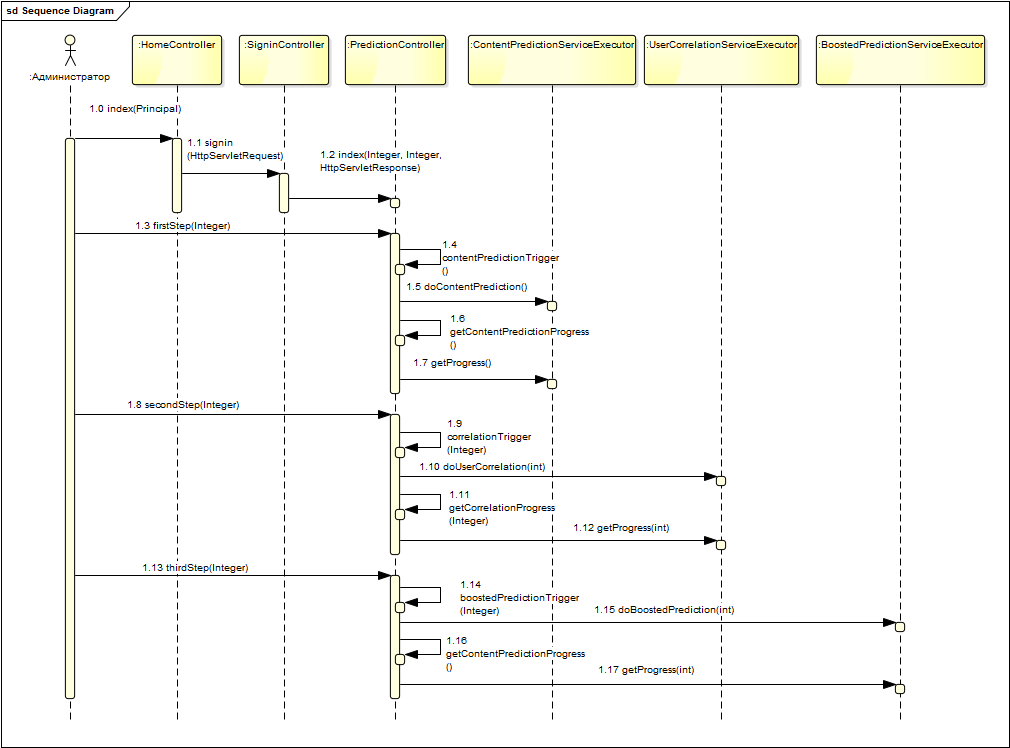


Рисунок 11 ­– Диаграмма последовательностей

|  |  |
| --- | --- |
| **ДИАГРАММА КОМПОНЕНТОВ** | **ПРИЛОЖЕНИЕ 9** |

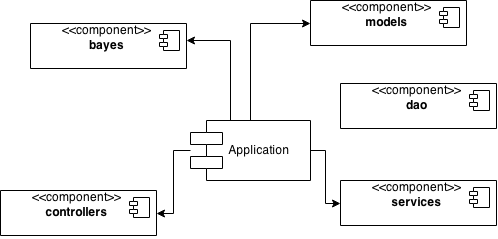


Рисунок 12 – Диаграмма компонентов

|  |  |
| --- | --- |
| **СХЕМА БАЗЫ ДАННЫХ** | **ПРИЛОЖЕНИЕ 10** |

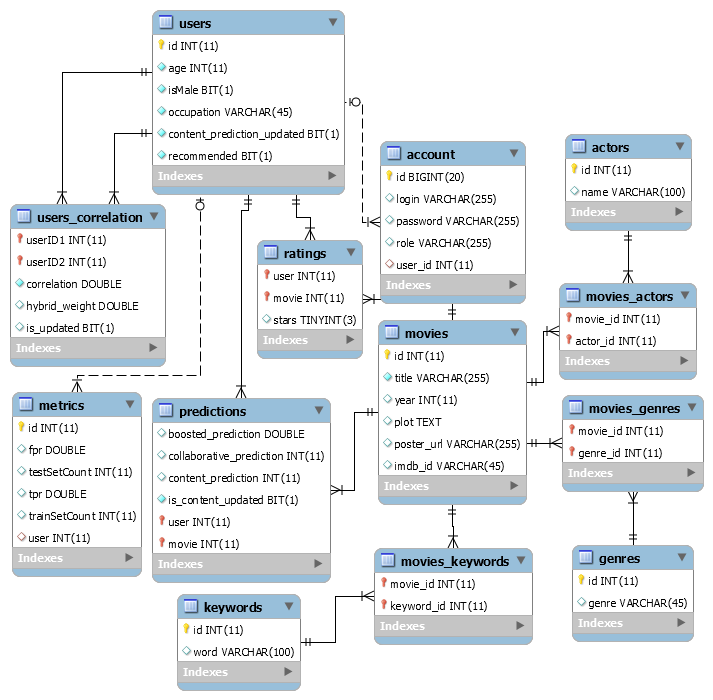


Рисунок 13 – Схема базы данных

**ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Лист регистрации изменений | | | | | | | | | |
| Номера листов (страниц) | | | | | Всего листов (страниц в докум.) | № документа | Входящий № сопрово-дительно-  го докум. и дата | Подл. | Да-  та |
| Изм. | Изменен-  ных | Заменен-  ных | новых | аннулированных |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |