Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования «Пермский государственный аграрно-технологический университет имени академика Д.Н. Прянишникова»

Кафедра Информационных технологий

и программной инженерии

**ОТЧЁТ ПО УЧЕБНОЙ ПРАКТИКЕ**

(проектно-технологическая практика)

на тему: «Разработка информационной системы

для автоматизации бизнес-процесса»

Выполнил:

студент группы ПИб\_-\_\_\_\_\_-\_\_\_\_\_

направления подготовки

09.03.03 Прикладная информатика

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

Проверил:

доцент кафедры ИТиПИ, к.т.н., доцент

Беляков Андрей Юрьевич

Пермь – 202\_\_

Содержание

[Постановка задачи на проектирование 3](#_Toc108878276)

[Анализ технологий проектирования 4](#_Toc108878277)

[Реализация функционала информационной системы 8](#_Toc108878278)

[Заключение 12](#_Toc108878279)

[Список источников 13](#_Toc108878280)

[ПРИЛОЖЕНИЕ А 14](#_Toc108878281)

[ПРИЛОЖЕНИЕ Б 20](#_Toc108878282)

# Постановка задачи на проектирование

Программа автоматизирует запись отзывов и сортировку их по убыванию оценки. Данной программой будут пользователи любители кино, чтобы оставить отзыв на фильм или прочитать отзывы на фильм, который хотят посмотреть.

Программа должна считывать и записывать отзывы о фильме в базу данных.

Интерфейс программы должен быть интуитивно понятен, чтобы любой человек смог написать отзыв.

Программа должна иметь защиту от DDOS-атак. Также должна быть защита базы данных от возможных взломов.

Программа должна быть кроссплатформенной, а именно запускаться на любом устройстве с любой операционной системой.

# Анализ технологий проектирования

**SQLite** — это встраиваемая кроссплатформенная БД, которая поддерживает достаточно полный набор команд SQL и доступна в исходных кодах (на языке C).

Исходные коды SQLite находятся в public domain, то есть вообще никаких ограничений на использование.

SQLite - везде. Как минимум, на любом смартфоне.

Она очень надежна. При выпуске версии она проходит через ряд серьезнейших автоматических тестов (проводится ~ 2 млн тестов), покрытие кода тестами 100% (с августа 2009).

SQLite поддерживает динамическое типизирование данных. Возможные типы значений: INTEGER, REAL, TEXT и BLOB. Также поддерживается специальное значение NULL.

Размеры значений типа TEXT и BLOB не ограничены ничем, кроме константы SQLITE\_MAX\_LENGTH в исходниках sqlite, равной миллиарду.

Каждое значение в любом поле любой записи может быть любого из этих типов, независимо от типа, указанного при объявлении полей таблицы. Указанный при объявлении поля тип хранится для справки в его исходном написании, и используется в качестве основы для выбора предпочтений (так называемое «type affinity»: это подход, редко встречающийся в других СУБД) при выполнении неявных преобразований типов на основании похожести этого названия типа на что-либо, знакомое SQLite. В этот алгоритм зашит обширный перечень практикуемых в других СУБД вариантов названий типов данных. Если безопасного преобразования записываемого значения в предпочитаемый тип не получается, SQLite записывает значение в его исходном виде. Для получения значений из базы есть ряд функций для каждого из типов, и, если тип хранимого значения не соответствует запрашиваемому, оно тоже, по возможности, преобразуется.

Сама библиотека SQLite написана на C; существует большое количество привязок к другим языкам программирования, в том числе Apple Swift, Delphi, C++, Java, C#, VB.NET, Python, Perl, Node.js, PHP, PureBasic, Tcl (средства для работы с Tcl включены в комплект поставки SQLite), Ruby, Haskell, Scheme, Smalltalk, Lua и Parser, а также ко многим другим. Полный список существующих средств размещён на странице проекта.

**MySQL** — свободная реляционная система управления базами данных. Разработку и поддержку MySQL осуществляет корпорация Oracle, получившая права на торговую марку вместе с поглощённой Sun Microsystems, которая ранее приобрела шведскую компанию MySQL AB. Продукт распространяется как под GNU General Public License, так и под собственной коммерческой лицензией. Помимо этого, разработчики создают функциональность по заказу лицензионных пользователей. Именно благодаря такому заказу почти в самых ранних версиях появился механизм репликации.

MySQL имеет APIи коннекторы для языков Delphi, C, C++, Эйфель, Java, Лисп, Perl, PHP, Python, Ruby, Smalltalk, Компонентный Паскаль, Tcl и Lua, библиотеки для языков платформы .NET, а также обеспечивает поддержку для ODBC посредством ODBC-драйвера MyODBC.

MyODBC представляет собой драйвер ODBC (2.50) уровня 0 (с некоторыми возможностями уровней 1 и 2) для подсоединения совместимого с ODBC приложения к MySQL. MyODBC работает на всех системах Microsoft Windows и на большинстве платформ Unix.

**PostgreSQL**- свободная объектно-реляционная система управления базами данных (СУБД).

Существует в реализациях для множества UNIX-подобных платформ, включая AIX, различные BSD-системы, HP-UX, IRIX, Linux, macOS, Solaris/OpenSolaris, Tru64, QNX, а также для Microsoft Windows.

Сильными сторонами PostgreSQL считаются:

- высокопроизводительные и надёжные механизмы транзакций и репликации;

- расширяемая система встроенных языков программирования: в стандартной поставке поддерживаются PL/pgSQL, PL/Perl, PL/Python и PL/Tcl; дополнительно можно использовать PL/Java, PL/PHP, PL/Py, PL/R, PL/Ruby, PL/Scheme, PL/sh и PL/V8, а также имеется поддержка загрузки модулей расширения на языке C;

- наследование;

- возможность индексирования геометрических (в частности, географических) объектов и наличие базирующегося на ней расширения PostGIS;

- встроенная поддержка слабоструктурированных данных в формате JSON с возможностью их индексации;

- расширяемость (возможность создавать новые типы данных, типы индексов, языки программирования, модули расширения, подключать любые внешние источники данных).

Функции могут писаться с использованием одного из следующих языков:

- Встроенный процедурный язык PL/pgSQL, во многом аналогичный языку PL/SQL, используемому в СУБД Oracle;

- Скриптовые языки — PL/Lua, PL/LOLCODE, PL/Perl, PL/PHP, PL/Python, PL/Ruby, PL/sh, PL/Tcl, PL/Scheme, PL/v8 (Javascript);

Классические языки — C, C++, Java (через модуль PL/Java);

Статистический язык R (через модуль PL/R).

Я решил выбрать SQLite так как кроме компьютеров она может размещаться на смартфоне, а также имеется поддержка языка программирования Node.js.

**C#** — объектно-ориентированный язык программирования. Разработан в 1998—2001 годах группой инженеров компании Microsoft под руководством Андерса Хейлсберга и Скотта Вильтаумотакак язык разработки приложений для платформы Microsoft .NET Framework и .NET Core. Впоследствии был стандартизирован как ECMA-334 и ISO/IEC 23270.

С#‎ разрабатывался как язык программирования прикладного уровня для CLR и, как таковой, зависит, прежде всего, от возможностей самой CLR. Это касается, прежде всего, системы типов С#‎, которая отражает BCL. Присутствие или отсутствие тех или иных выразительных особенностей языка диктуется тем, может ли конкретная языковая особенность быть транслирована в соответствующие конструкции CLR. Так, с развитием CLR от версии 1.1 к 2.0 значительно обогатился и сам C#; подобного взаимодействия следует ожидать и в дальнейшем (однако, эта закономерность была нарушена с выходом C# 3.0, представляющего собой расширения языка, не опирающиеся на расширения платформы .NET). CLR предоставляет С#‎, как и всем другим .NET-ориентированным языкам, многие возможности, которых лишены «классические» языки программирования. Например, сборка мусора не реализована в самом C#‎, а производится CLR для программ, написанных на C# точно так же, как это делается для программ на VB.NET, J# и др.

**Python** — высокоуровневый язык программирования общего назначения с динамической строгой типизацией и автоматическим управлением памятью, ориентированный на повышение производительности разработчика, читаемости кода и его качества, а также на обеспечение переносимости написанных на нём программ. Язык является полностью объектно-ориентированным в том плане, что всё является объектами. Необычной особенностью языка является выделение блоков кода пробельными отступами. Синтаксис ядра языка минималистичен, за счёт чего на практике редко возникает необходимость обращаться к документации. Сам же язык известен как интерпретируемый и используется в том числе для написания скриптов. Недостатками языка являются зачастую более низкая скорость работы и более высокое потребление памяти написанных на нём программ по сравнению с аналогичным кодом, написанным на компилируемых языках, таких как C или C++.

На основе Python было создано несколько специализированных подмножеств языка, в основном предназначенных для статической компиляции в машинный код. Некоторые из них перечислены ниже.

RPython— созданная в рамках проекта PyPy сильно ограниченная реализация Python без динамизма времени исполнения и некоторых других возможностей. Код на RPython можно компилировать во множество других языков/платформ — C, JavaScript, Lisp, .NET, LLVM. На RPython написан интерпретатор PyPy.

Pyrex — ограниченная реализация Python, но несколько меньше, чем RPython. Pyrex расширен возможностями статической типизации типами из языка C и позволяет свободно смешивать типизированный и не типизированный код. Предназначен для написания модулей расширений, компилируется в код на языке C.

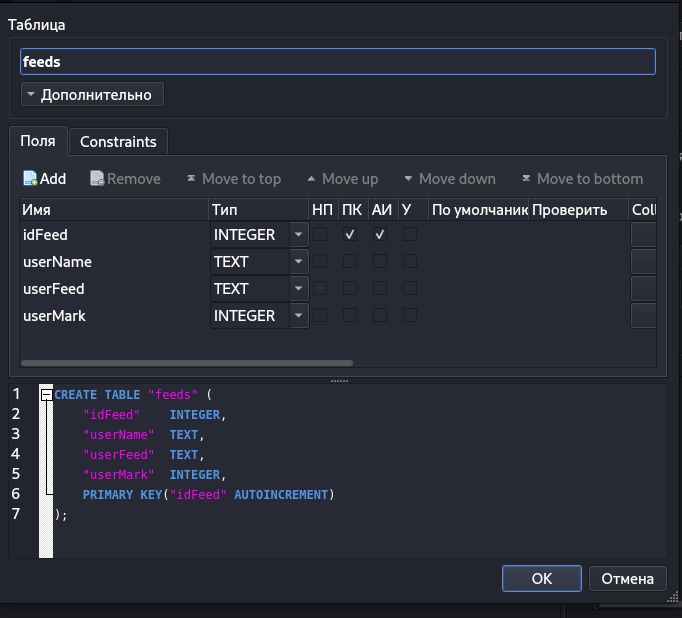
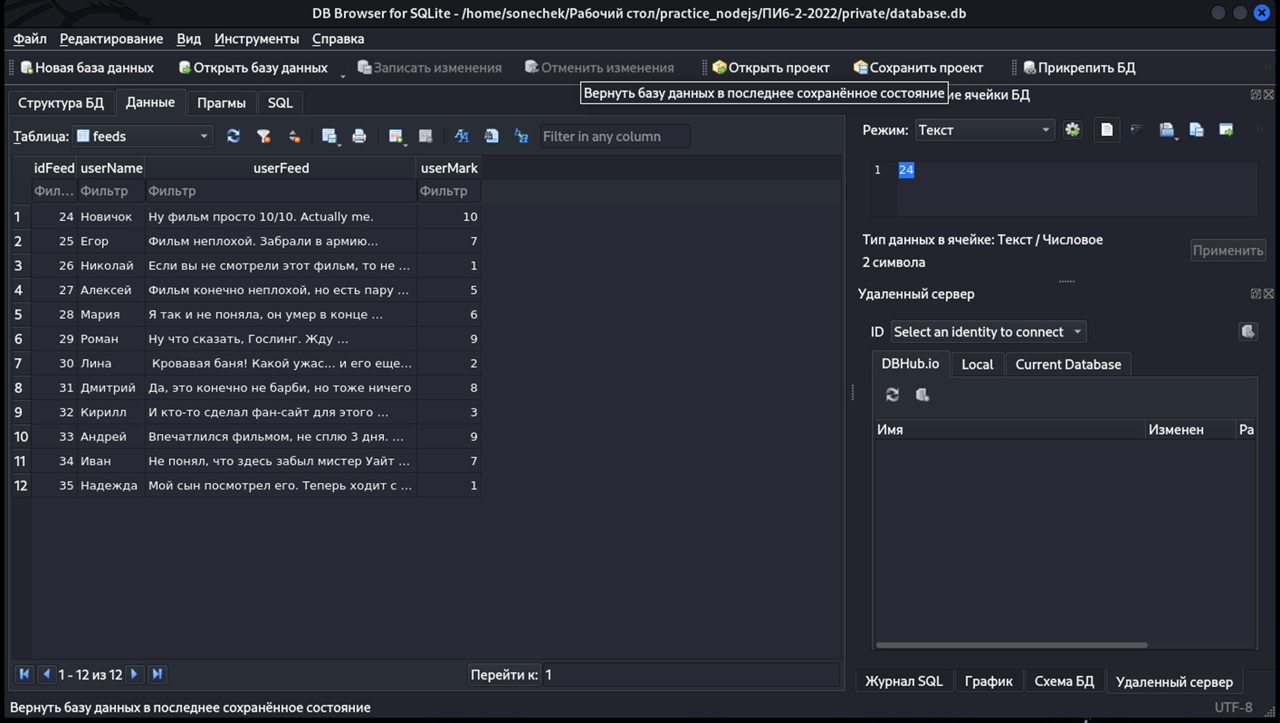
Cython — расширенная версия Pyrex.

Проект Shedskin — предназначен для компиляции неявно статически типизированного кода на Python в оптимизированный код на языке C++.

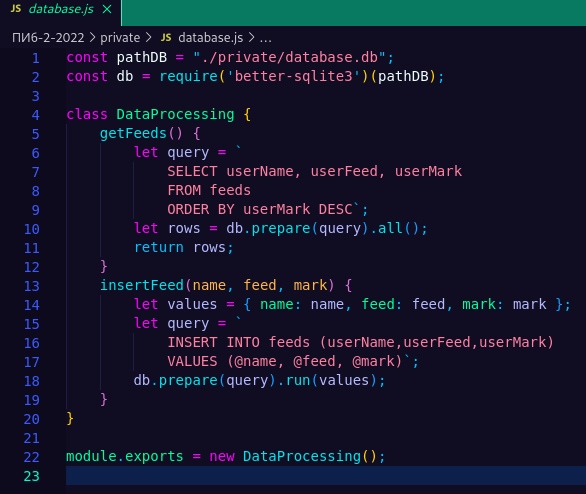
**Node** или **Node.js** — программная платформа, основанная на движке V8 (компилирующем JavaScript в машинный код), превращающая JavaScript из узкоспециализированного языка в язык общего назначения. Node.js добавляет возможность JavaScript взаимодействовать с устройствами ввода-вывода через свой API, написанный на C++, подключать другие внешние библиотеки, написанные на разных языках, обеспечивая вызовы к ним из JavaScript-кода. Node.js применяется преимущественно на сервере, выполняя роль веб-сервера, но есть возможность разрабатывать на Node.js и десктопные оконные приложения (при помощи NW.js, AppJS или Electron для Linux, Windows и macOS) и даже программировать микроконтроллеры (например, tessel, low.js и espruino). В основе Node.js лежит событийно-ориентированное и асинхронное (или реактивное) программирование с неблокирующим вводом/выводом.

Я решил выбрать Node.js так как им можно без проблем писать код как на Linux, так и на Windows. А также он поддерживается базой данных SQLite.

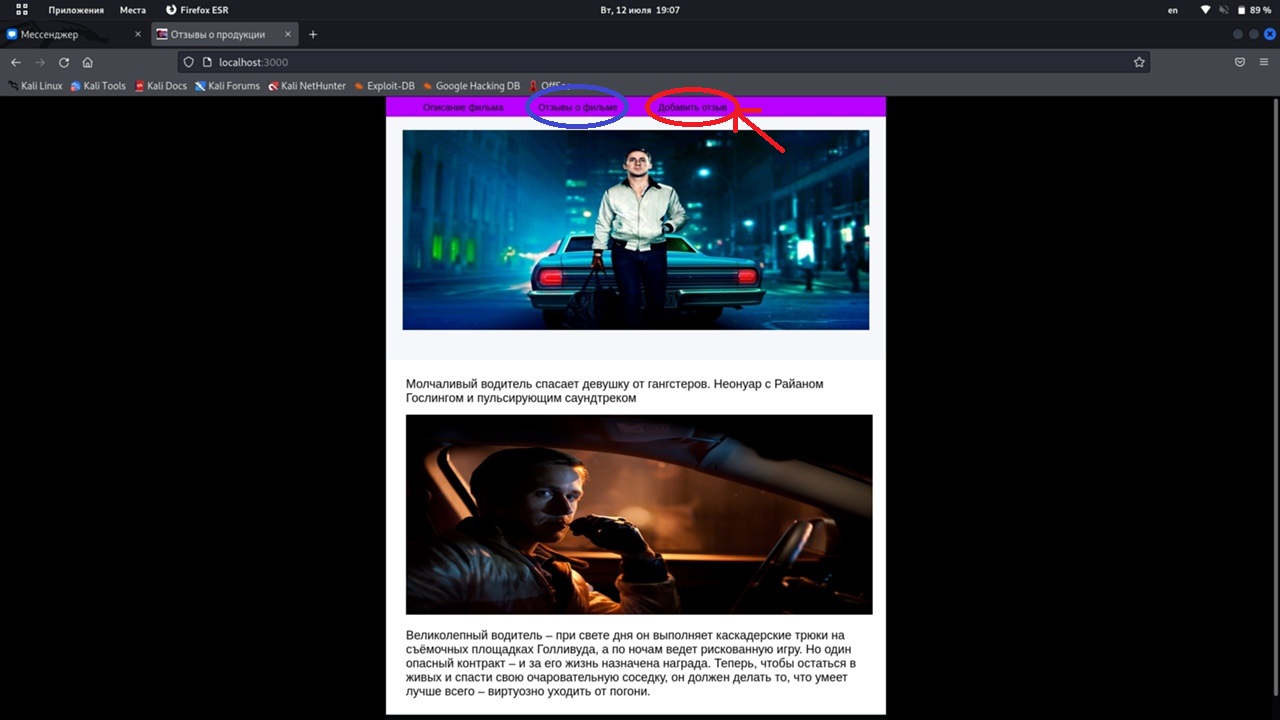
# Реализация функционала информационной системы



На рисунках показана модель базы данных, типы данных и информация, хранящаяся в ней.

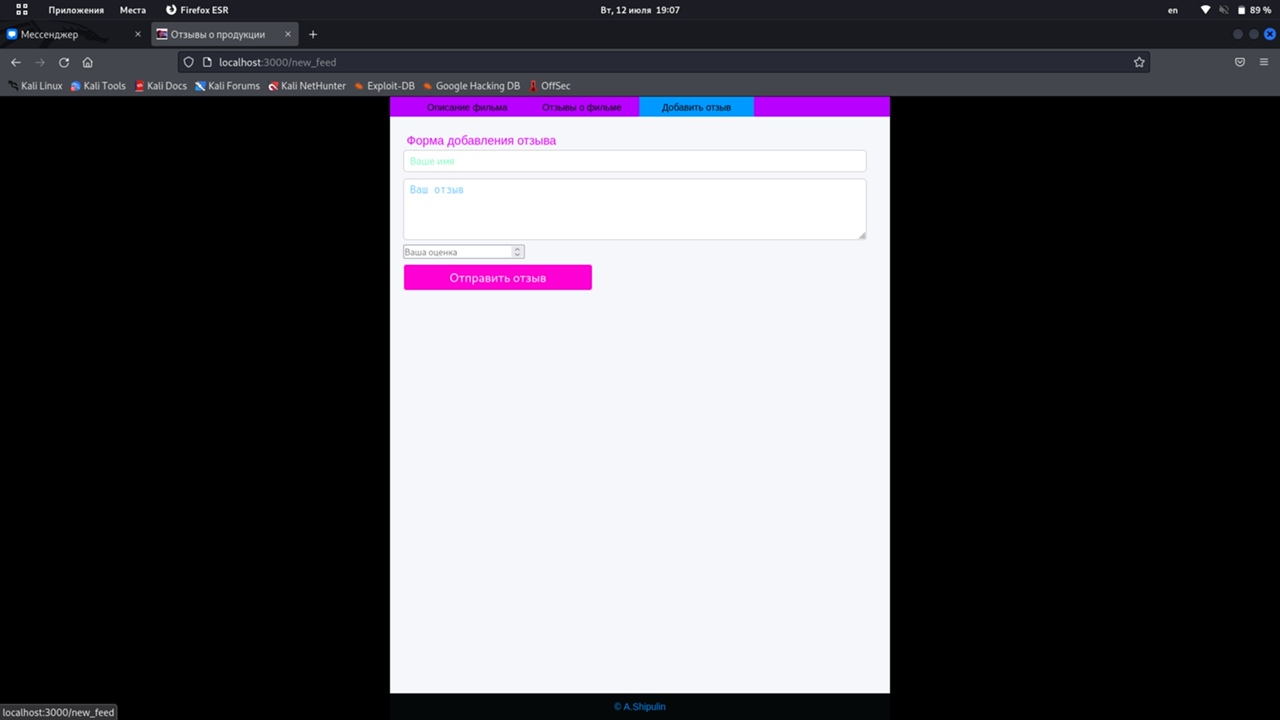


На рисунке указан SQL запрос из кода к базе данных, который выводит информацию из базы данных с сортировкой по оценке.

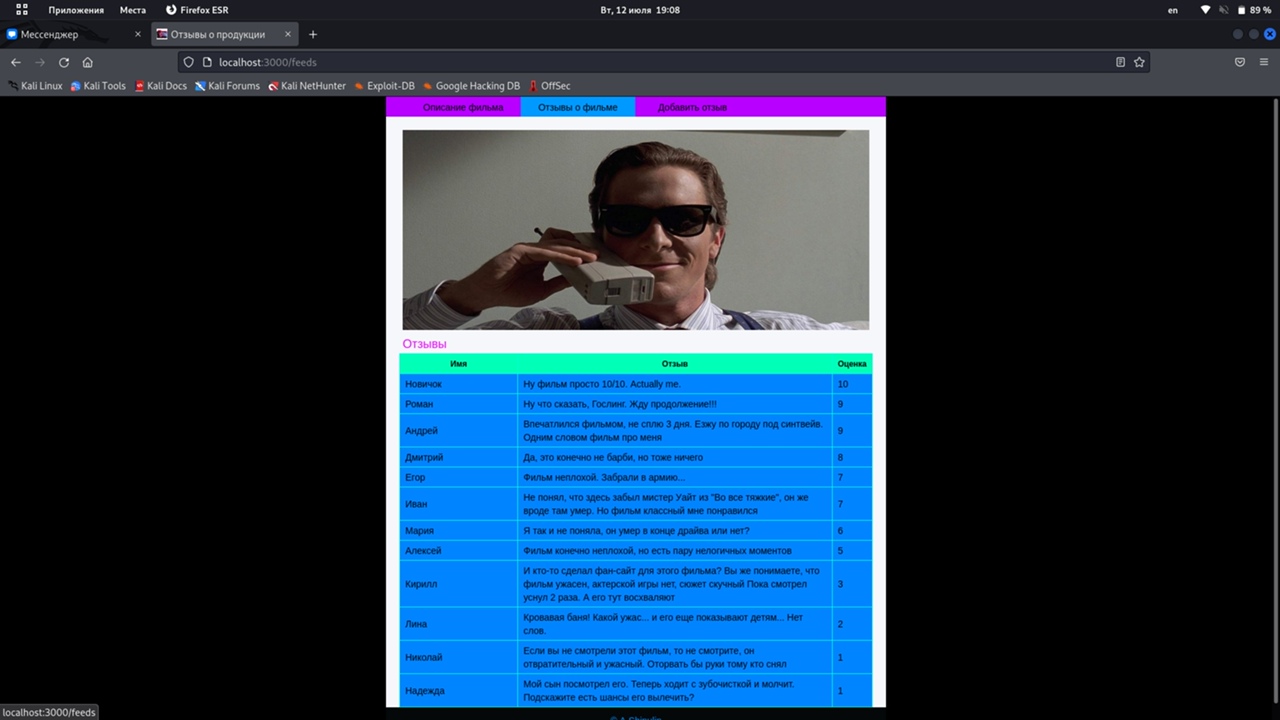


На рисунке красным кружком показана кнопка, на которую нужно нажать чтобы перейти на форму с написанием отзыва и синим кружком показана кнопка, на которую нужно нажать чтобы просмотреть написанные отзывы.

По умолчанию при заходе на сайт вы попадаете на главную страницу «Описание фильма».



На данном рисунке показа форма добавления отзыва, первая строчка — это добавление имени автора отзыва, вторая сам отзыв и третья это оценка фильма по 10-ти бальной шкале. После нажатия на кнопку отправить отзыв, отзыв будет добавлен в базу данных и выведен на страницу с отзывами о фильме.



На рисунке можно увидеть все отзывы, которые были написаны, имена авторов и их оценки, отсортированные в порядке убывания.

# Заключение

Были проанализированы несколько языков программирования (Python, C#, Node.js) и несколько СУБД (MySQL, SQLite, PostgreSQL), после чего был выбран нужный язык программирования и СУБД для создания нужного функционала.

Была спроектирована база данных и программа, которая принимает отзывы о фильме. Реализованы функции сортировки данных в базе данных, а также получение оценки пользователя программы.

Эффективность автоматизации высока, так как данные автоматически получаются и автоматически попадают на вкладку отзывов тем самым ручной работы здесь не имеется.

В дальнейшем можно реализовать функцию аккаунтов, чтобы человек мог написать только один отзыв о фильме. Также можно добавить на сайт функцию поиска фильма на стриминг-площадках таких как Кинопоиск, ivi, Amediateka и др. Расширить количество фильмов на сайте, сделать удобный поиск по фильмам, а также собрать подборки фильмов по интересам пользователя.

# Список источников

1. SQLite — замечательная встраиваемая БД (часть 1) [Электронный ресурс] Режим доступа: https://habr.com/ru/post/149356/

2. Node.js [Электронный ресурс] Режим доступа: https://ru.wikipedia.org/wiki/Node.js#Версии

3. Python [Электронный ресурс] Режим доступа: https://ru.wikipedia.org/wiki/Python#Реализации

4. PostgreSQL [Электронный ресурс] Режим доступа: https://ru.wikipedia.org/wiki/PostgreSQL

5. C# [Электронный ресурс] Режим доступа: https://ru.wikipedia.org/wiki/C\_Sharp

6. MySQL [Электронный ресурс] Режим доступа: https://ru.wikipedia.org/wiki/MySQL

7. SQLite [Электронный ресурс] Режим доступа: https://ru.wikipedia.org/wiki/SQLite

8.Как работают веб-приложения [Электронный ресурс] Режим доступа: https://habr.com/ru/post/450282/

9. How to choose a Technology Stack for Web Application Development? – GeeksforGeeks. [Электронный ресурс] Режим доступа: https://www.geeksforgeeks.org/how-to-choose-a-technology-stack-for-web-application-development/

ПРИЛОЖЕНИЕ А

наименование организации – разработчика ТЗ на АС

УТВЕРЖДАЮ

Руководитель

(должность, наименование предприятия – заказчика АС)

Личная подпись Расшифровка подписи

(печать)

Дата

УТВЕРЖДАЮ

Руководитель

(должность, наименование предприятия – разработчик АС)

Личная подпись Расшифровка подписи

(печать)

Дата

наименование вида АС

наименование объекта автоматизации

сокращённое наименование АС

**ТЕХНИЧЕСКОЕ ЗАДАНИЕ**

На листах

Действует с

2022

1. Общие сведения
2. Наименование системы

Полное наименование разрабатываемой системы – «Автоматизированное веб-приложение «Отзыв о фильме».

Краткое наименование – «Веб-приложение «Отзыв о фильме».

1. Наименование заказчика и исполнителя

Организация: ФГБОУ ВО Пермский ГАТУ.

Адрес: ул. Петропавловская, 23.

Телефон: +7 (342) 217-90-66;

Исполнитель: Шипулин Александр Владимирович.

1. Плановые сроки начала и окончания работ

Дата начала работ: 05.07.2021.

Дата окончания работ: 16.07.2021.

1. Назначение и цели создания системы

К целям создания веб-приложения «Отзыв о фильме" можно отнести:

* сбор отзывов и аналитика их;
* автоматизировать сбор и сортировку отзывов о фильме.

Достижение целей приведёт к следующим положительным результатам:

* режиссер сможет узнать, что конкретно не понравилось зрителям и устранение недочетов в его следующих фильмах;
* зритель сможет узнать стоит ли тратить время на этот фильм или нет.

1. Характеристика объекта автоматизации

Объектом автоматизации является веб-приложение.

1. Требования к системе

Общие требования к веб-приложению «Отзыв о фильме» являются:

* надёжность и работоспособность;
* интуитивно понятный интерфейс;
* соблюдение информационной безопасности и разграничение прав доступа к данным.

1. Требования к способам и средствам связи для информационного обмена между компонентами

Для обеспечения информационного обмена компоненты подсистемы должны взаимодействовать с объединённой информационной базой данных. Благодаря хранению данных в различных схемах, таких как оценка пользователя или отзыв пользователя веб-приложение «Отзыв о фильме» может объединить эти данные представив их как единый информационный поток.

1. Перспективы развития, модернизация системы

Дальнейшим развитием веб-приложения «Отзыв о фильме» может быть расширение количества фильмов, рекомендации фильмов пользователям по их интересам.

1. Требования к квалификации персонала и режиму его работы

Для обеспечения максимальной работоспособности пользователей должны устанавливаться перерывы:

* через 2 часа после начала смены и через 1,5–2 часа продолжительностью 15 минут;
* через каждый час работы продолжительностью 10 минут

Для эксплуатации веб-приложения «Отзыв о фильме» определены следующие роли:

* системный администратор – должен быть квалифицированным специалистом с практическим опытом выполнения работ по администрированию программных и технических средств. В обязанности входит: установка, модернизация, настройка программного обеспечения, ведение учётных записей портала;
* администратор баз данных – должен быть квалифицированным специалистом с практическим опытом выполнения работ по администрированию СУБД, проектированию БД, оптимизации производительности, разграничению прав и ролей, а также резервного копирования и обеспечение целостности БД;
* пользователь веб-приложения – должен иметь опыт работы с персональным компьютером на уровне опытного пользователя и свободно осуществлять базовые действия с веб-приложением посредством браузера с доступом в интернет.

1. Требования к надёжности технических средств и программного обеспечения

Надёжность по отношению к техническим средствам должна обеспечиваться использованием в системе средств повышенной отказоустойчивости и их резервированием, а также дублированием носителей информационных банков данных.

Надёжность программного комплекса обеспечивается использованием сертифицированных операционных систем, общесистемных программных средств и инструментальных программных систем, используемых при разработке программного обеспечения. Само программное обеспечение должно обеспечивать защиту от некорректных действий пользователей и ошибочных исходных данных.

1. Требования к безопасности

Разрабатываемое веб-приложение «Отзыв о фильме» должно обеспечивать безопасный доступ к данным, предотвращая несанкционированный доступ или модифицирование данных. Модуль аутентификации должен обеспечивать защищённый доступ ко всему программному интерфейсу приложения.

1. Требования по эргономике и технической эстетике

Модуль должен иметь удобный и интуитивно понятный графический пользовательский интерфейс

Пользовательский интерфейс модуля также должен аккомпанировать цветовой гамме и общему стилю фильма.

1. Требования к программному обеспечению

При проектировании веб-приложения «Отзыв о фильме» необходимо эффективно использовать веб-фреймворк Vue.js, в качестве серверного окружения используется программная платформа Node.js, а для хранения данных применяется СУБД SQLite.

1. Требования к техническому обеспечению

Техническое обеспечение системы должно максимально и наиболее эффективно использовать существующее в отделе автоматизации оборудование:

* процессор – 2х Intel Xeon 3.7 ГГц;
* оперативная память – 32 ГБ;
* дисковая система – 2 х 1ТБ;
* сетевой адаптер – 1 Гб/с.

1. Порядок контроля и приёмки системы

Приёмо-сдаточные испытания системы проводятся с привлечением сотрудников отдела автоматизации. По результатам опытной эксплуатации оформляется акт о приёме работ. Акт содержит заключение о соответствии системы техническому заданию.

1. Требования к составу и содержанию работ подготовки объекта автоматизации к вводу системы в действие

При подготовке к вводу в эксплуатацию веб-приложения «Отзыв о фильме» отдел управления информатизации должен обеспечить выполнение следующих работ:

* определить подразделение и ответственных должностных лиц для внедрения веб-приложения;
* обеспечить пользователей руководством, которое поможет быстрее освоить внедрённое веб-приложение;
* провести опытную эксплуатацию веб-приложения «Отзыв о фильме».

# ПРИЛОЖЕНИЕ Б

**Листинг кода - m\_feeds.js**

**const** model = {

title: "Отзывы о фильме",

feeds: []

};

module.exports = { model };

**Листинг кода – m\_intro.js**

**const** model = {

title: "Отзывы о фильме",

};

module.exports = { model };l };

**Листинг кода – m\_new\_feed.js**

**const** model = {

title: "Отзывы о фильме",

tableName: "Форма добавления отзыва"

};

module.exports = { model };

**Листинг кода – database.js**

**const** pathDB = "./private/database.db";

**const** db = require('better-sqlite3')(pathDB);

class DataProcessing {

getFeeds() {

let query = `

SELECT userName, userFeed, userMark

FROM feeds

ORDER BY userMark DESC`;

let rows = db.prepare(query).all();

**return** rows;

}

insertFeed(name, feed, mark) {

let values = { name: name, feed: feed, mark: mark };

let query = `

INSERT INTO feeds (userName,userFeed,userMark)

VALUES (@name, @feed, @mark)`;

db.prepare(query).run(values);

}

}

module.exports = **new** DataProcessing();

**Листинг кода – r\_feeds.js**

**const** express = require('express');

**const** router = express.Router();

**const** { model } = require("../models/m\_feeds.js");

**const** dp = require("../private/database.js");

router.**get**('/', (req, res) => {

model.feeds = dp.getFeeds();

res.render('feeds.ejs', model); *// render view*

});

module.exports = router;

**Листинг кода – r\_intro.js**

**const** express = require('express');

**const** router = express.Router();

let { model } = require('../models/m\_intro.js');

router.**get**('/', (req, res) => {

res.render('intro.ejs', model); *// render view*

});

module.exports = router;

**Листинг кода – r\_new\_feed.js**

**const** express = require('express');

**const** router = express.Router();

let { model } = require("../models/m\_new\_feed.js");

**const** dp = require("../private/database.js");

**const** htmlParser = express.urlencoded({extended: **false**});

router.**get**('/', (req, res) => {

res.render('new\_feed.ejs', model); *// render view*

});

router.post('/', htmlParser, (req, res) => {

let { userName, userFeed, userMark } = req.body;

dp.insertFeed(userName, userFeed, userMark);

res.redirect("/feeds");

});

module.exports = router;

**Листинг кода – app.js**

**const** express = require('express');

**const** app = express();

app.use('/public', express.**static**(\_\_dirname + '/public'));

app.**set**('view engine', 'ejs');

**const** r\_intro = require('./routes/r\_intro.js');

**const** r\_feeds = require('./routes/r\_feeds.js');

**const** r\_new\_feed = require('./routes/r\_new\_feed.js');

app.use(['/intro','/'], r\_intro);

app.use('/feeds', r\_feeds);

app.use('/new\_feed', r\_new\_feed);

app.listen(3000, "localhost", () => console.log(`--> to stop: Ctrl+C`));

**Листинг кода – index.js**

**const** express = require('express');

**const** app = express();

app.use('/public', express.**static**(\_\_dirname + '/public'));

app.**get**("/", (req, res) => {

let pagePath = \_\_dirname + "/views/" + "intro.html";

res.sendFile(pagePath);

});

app.**get**("/:page", (req, res) => {

let page = req.params.page;

let pagePath = \_\_dirname + "/views/" + page;

res.sendFile(pagePath);

});

app.listen(3000, "localhost", () => console.log(`--> to stop: Ctrl+C`));