Яндекс.Лицей

Реферат на тему:

«Разработка интернет сайта для моделирования работы системы дистанционного бронирования билетов в кинотеатр»

Над проектом работали:

Коршунов Анатолий,

Олег Ролдугин,

Ксения Бондаренко

Преподаватель:

Косов А.В

Волгоград 2021

**СОДЕРЖАНИЕ**

Введение

1. Обоснование выбора среды разработки
   1. Обзор аналогов
   2. Анализ и декомпозиция предметной области
2. Проектирование
   1. Описание некоторых алгоритмов
   2. Описание реализации
3. Описание программы для пользователя
4. Заключение

Список используемых источников

Приложение

**ВВЕДЕНИЕ**

Цель проекта — разработать сайт в сети интернет, который предоставит пользователю возможность ознакомиться с фильмами, которые находятся в прокате кинотеатрах, а также оформить бронь одного или нескольких мест на сеанс выбранного кинофильма.

Процесс работы проходил в 4 основных этапа:

1. Исследование предметной области.
2. Постановка и анализ задач.
3. Проектирование.
4. Разработка.

В результате было разработан сайт «FilmCenter», соответствующий поставленным целям.

Пояснительная записка к курсовому проекту была выполнена в текстовом редакторе Microsoft Office Word 2010 и представлена в печатном виде.

|  |
| --- |
| ЗАДАНИЕ |

1. Тема проекта: Разработка интернет сайта для моделирования работы системы дистанционного бронирования билетов в кинотеатр .

2. Срок сдачи проекта: 24.04.2021

Программа состоит из:

* Пользовательского интерфейса
* Администраторского интерфейса

Программа имеет функции:

* Показ доступных фильмов.
* Поиск интересующего фильма через возможности встроенного фильтра и поисковую строку.
* Просмотр описания, трейлеров, а также изображений фильмов
* Заказ билетов на фильм
* Отправка билетов на почту.
* Добавление новых фильмов и редактирование старых.
* Ручное и автоматическое создание сессий для фильмов.

Проект представляет собой сайт, в котором пользователь может ознакомиться с доступными фильмами и выбрать интересующий его фильм или подобрать фильм по нескольким критериям (год, жанр, продолжительность, режиссер). Выбрав фильм, пользователь может узнать краткое описание и жанр, длительность, год премьеры, а также актеров, принимавших участие в фильме, и режиссера. Кроме этого пользователь может ознакомиться с несколькими кадрами из фильма и трейлером к фильму. Ознакомившись, пользователь может забронировать билет. Для этого необходимо выбрать удобную дату, доступное время на сеанс и место в зале. Перед подтверждением брони пользователю необходимо ввести адрес электронной почты, на которую будут высланы билеты. Далее пользователь может, как заказать билеты на другое время или дату, так и вернуться к списку всех доступных фильмов в прокате.

Также проект подразумевает и работу администратора с базой данных. У администратора имеется доступ ко всем использующимся таблицам в проекте. Администратор может изменять, добавлять новые и удалять старые элементы. Кроме этого администратор имеет возможность сформировать новое расписание на неделю вперед или же вручную добавить сеанс в базу.

**1. ОБОСНОВАНИЕ ВЫБОРА СРЕДЫ РАЗРАБОТКИ**

Так как программа курсового проекта должна быть реализована на языке программирования Python, то было принято решение использовать среду разработки Pycharm. Данная среда является мощным и удобным инструментом в разработке программ на языке программирования Python.

Сайт разрабатывался с использованием библиотеки Flask, позволяющей создать сервер для обработки url запросов и настроить корректную работу с апи.

Разработанный сайт был размещен на платформе Heroku, предоставляющая удаленно запускать различные приложения, и получать доступ через url ссылки. Платформа работает с PostgreSQL бд и предоставляет доступ к удаленной базе. Создана база данных была с использованием библиотеки Sqlalchemy. Добавление новых элементов осуществлялось как с использованием разработанного интерфейса администратора, так и с использованием приложения pgAdmin4, осуществляющего подключение к удаленной базе и предоставляющее инструменты для внесения изменений в бд.

**1.1 ОБЗОР АНАЛОГОВ**

Разработанный сайт имеет аналоги. Рассмотрим один из них. Сайт <http://xn--c1aksadab.xn--p1ai/kinoteatr/> реализует аналогичные функции предоставления расписания на доступные фильмы в кинотеатре. Главное окно сайта-аналога представлено на рисунке 2.1.

|  |
| --- |
|  |
| Рисунок 2.1 — Главное окно сайта-аналога |

Главными недостатками данного сайта является:

Отсутствие подробного ознакомления с фильмами, находящимися в прокате.

Отсутствие системы бронирования мест и получения билетов на почту.

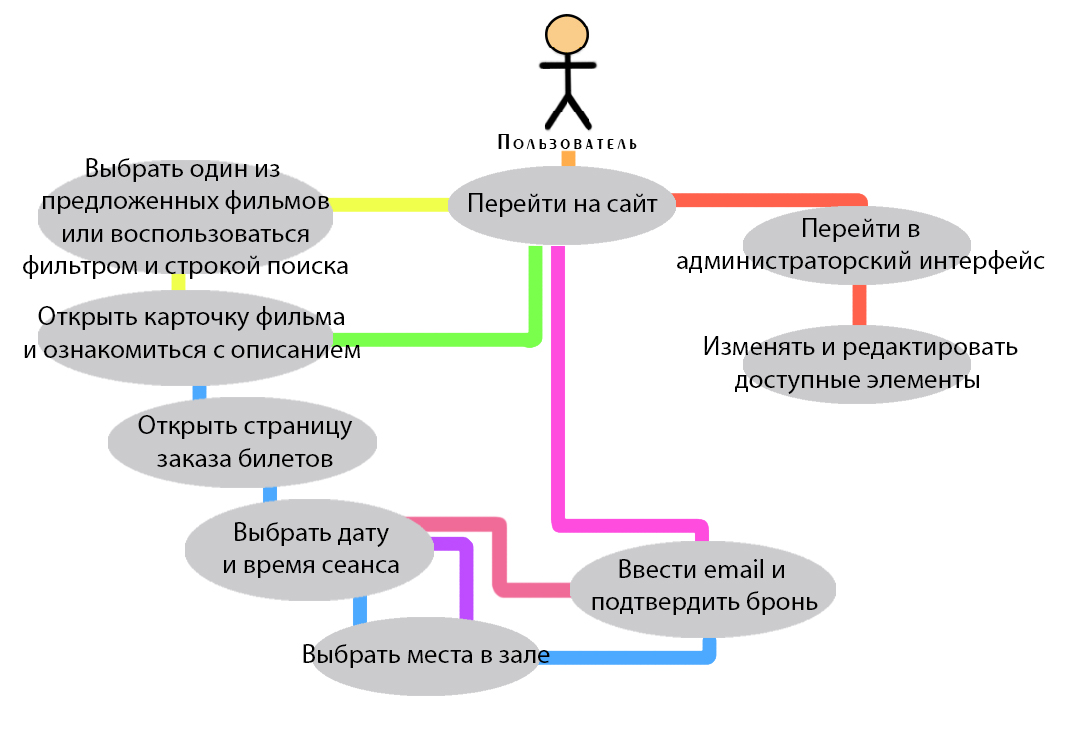
**1.2 АНАЛИЗ И ДЕКОМПОЗИЦИЯ ПРЕДМЕТНОЙ ОБЛАСТИ**

Исходя из требований к программе, были выделены основные классы:

* data.films.Film – класс, представляющий структуру таблицы films (фильмы) в бд.
* data.images.Image – класс, представляющий структуру таблицы images (картинок к фильмам) в бд.
* data.film\_sessions.FilmSession - класс, представляющий структуру таблицы film\_session (сеансов на фильмы) в бд.
* data.places.Place – класс, представляющий структуру таблицы place (заказанных мест) в бд.
* data.associations.Genre – класс, представляющий собой структуру таблицы genres (жанров) в бд.
* data.admins.AdminRole – класс, представляющий структуру таблицы админов в бд.
* api.films\_resource.FilmResource – класс апи, предоставляющий инструменты для получения определенных фильмов из бд, а также удаления и изменения существующих фильмов.
* api.films\_resource.FilmListResource – класс апи, предоставляющий инструменты для получения списка существующих фильмов и добавления новых фильмов.
* api.film\_session\_resource.FilmSessionResource - класс апи, предоставляющий инструменты для получения определенных сессий на фильм из бд, а также удаления и изменения существующих сессий.
* api.film\_session\_resource.FilmSessionListResource - класс апи, предоставляющий инструменты для получения списка существующих сессий на фильмы и добавления новых сессий.
* forms.admin.LoginForm – класс, представляющий форму для логина администратора.
* modules.tiket\_PIL.Ticket – класс, создающий новый билет и возвращающий картинку в байтах.

Для того чтобы более точно определить последовательности действий, которые может осуществлять система в ответ на внешние воздействия пользователей была составлена диаграмма вариантов использования.  
Диаграмма вариантов использования представлена на рисунке 2.2.

Рисунок 2.2 — Диаграмма вариантов использования



**2. ПРОЕКТИРОВАНИЕ**

## 2.1. Описание некоторых алгоритмов

Алгоритм подключения к базе данных.

В проекте реализовано подключение к бд с помощью библиотеки Sqlalchemy. Данная библиотека позволяет подключиться к удаленной бд и создавать сессии для каких-либо изменений в бд.

Т.к. в нашем проекте использована система автоматического очищения каждые 24 часа, то появилась проблема с незакрытыми сессиями. Закрывать сессии можно как явно, вызывая метод close() у сессии, так и неявно, используя оператор with при создании сессии, который автоматически закроет сессию после завершения работы с бд.

Итак, порядок работы с бд можно описать по следующим пунктам:  
1) При создании приложения

1.1) Создается движок sqlalchemy.create\_engine для определенного типа бд. В нашем случае для PostgerSQL. Ссылка на PostgerSQL бд выглядит примерно так: postgresql://username:password@host:port/database\_name .

1.2) Создается sessionmaker, объект, который позволяет создавать сессии для работы с бд.

2) При работе приложения

2.1)Или с помощью оператора with создается сессия или сессия создается автоматически перед тем, как начнет выполняться код, на которую вела ссылка.

2.1.1) С помощью декоратора Flask before\_request перехватывается вызов функции ссылки. Далее создается сессия с бд и объект помещается с параметр Flask.g – позволяющий переносить какие-либо объекты между областями видимости функций.

2.1.2) С помощью декоратора Flask after\_request перехватывается событие завершения выполнения кода ссылкы. Тут и закрывается сессия с бд с помощью вызова метода close() у активной сессии.

2.2) Выполняется любая задача, требующая соединения с бд. После чего коннект с бд закрывается автоматически или необходимо вызвать метод close() у активной сессии, если не был использованы варианты соединения с бд, описанные в пунктах 2.1.1 и 2.1.2.

Алгоритм создания обработки запросов от пользователя:

1) Обычные ссылки перехватываются Flask декоратором route, который позволяет соединить ссылку с функцией, обрабатывающей эту ссылку.

2) Ссылки, ведущий на апи к нашему сайту реализованы через архитектурный стиль программирования RESTful. Перехваткой запросов от пользователя занимается внутренняя реализация библиотеки flask\_restful. От нас требуется лишь создать какой либо класс ресурса и зарегистрировать его с помощью вызова Flask. add\_resource.

Кроме библиотеки flask\_restful, также был использован класс Flask. Blueprint, который также позволяется перехватывать события от пользователя через декоратор route.

Алгоритм загрузки данных в шаблоны.

В проекте использован шаблонизатор Jinja. При загрузке шаблона на html через python с использованием render\_template(args), в функцию добавляются аргументы, к которым впоследствии можно будет обращаться через синтаксис Jinja. Также Jinja предоставляет огромный набор инструментов для работы с любыми объектами python.

Алгоритм создания билетов и отправка их на почту:

1) Из шаблона через POST запрос мы получаем данные о заказанных билетах.

2) Функция make\_order() занимается обработкой данных, вызывает функцию create\_places(), а также вызывает функцию send\_mail().

3) Функция create\_places() перебирает список с выбранными местами и создает объект Place, в котором хранится информация о месте, сеансе, а также уникальный код, который позволяет идентифицировать заказанный билет.

4) Функция send\_mail() запускается в отдельном потоке с использованием библиотеки threading. Функция формирует объект письма, создает картинки билетов с использованием класса Ticket, который возвращает картинку в байтах. Далее созданные билеты добавляются во вложения к письму.

5) После подготовки объекта письма оно отправляется на указанный адрес с помощью библиотеки smtplib.

Зачастую, такие сервисы почт, как Yandex.Mail, распознают наше письмо как спам. В результате оно оказывается в папке спама. И чтобы предупредить пользователя, после заказа билета, на странице всплывает подсказка о том, что билеты отправлены, и могут оказаться в спаме.

Также хочется отметить и работу класса Ticket, который формирует картинку билета с использованием библиотеки PIL. Данный класс открывает шаблон билета, и заполняет его переданными данными. Также при заполнении используется библиотека qrcode, которая позволяет сформировать картинку и отобразить ее через инструменты PIL.

**2.2. Описание реализации**

Реализация программы проходила на основе библиотек Flask, flask\_resful, flask\_admin, Sqlalchemy.

В ходе разработки программы было принято решение реализовать ее в виде проекта с открытым исходным кодом. Так же было принято решение использовать систему контроля версий. В связи с этим было решено размещать проект на сайте <https://github.com>.

Разрабатываемый программный продукт представляет собой многостраничный сайт.

Для обработки пользовательских запросов были применены декораторы flask.route, который перехватывают запрос и вызывают ту или иную функцию. Также для добавления апи к нашему проекту, были использованы библиотеки flask\_restful и Flask.Blueprint. Перечисленные библиотеки позволяют реализовать апи взаимодействия с данными в нашем проекте. Реализованы апи как для получения, удаления и изменения какого-то конкретного фильма или сессии на фильм, получения списка фильмов и сессий, так и для получения списка популярных фильмов и новых фильмов (тех, у кого дата премьеры была в текущем месяце). Также доступно апи для поиска фильмов по определенным параметрам (году, жанрам, продолжительности, названию, режиссеру).

Результат апи возвращается в формате json, с помощью вызова функции jsonify. Данная функция формирует json объект, используя сериализацию классов данных бд (которая наследуется от SerializerMixin).

Работа апи основана на библиотеке Sqlalchemy. Библиотека позволяет подключаться к базам данных и получать какие-либо данные или добавлять и изменять существующие. В проекте соединения с базой происходит при вызове create\_engine (). Последующие подключения происходят при вызове объекта sessionmaker. Объект создает подключение к бд, и с помощью query(), и filter() позволяет получать различные типы, фильтруемые по различным критериям.

Создание таблиц в бд происходил посредством формирования класса, который передается в create\_engine. Подобный класс должен наследоваться от SqlAlchemyBase. Чтобы класс был полноценной таблицей, ему нужно задать параметры, которые станут колонками в бд. Создание подобных параметров происходит с помощью sqlalchemy.Column(), куда передается какой тип данных будет хранить колонка и различные другие парамтеры.

Отрисовка шаблонов происходит с помощью вызова render\_template(). В функцию передается шаблон, который необходимо отобразить, и данные, которые будут переданы в шаблон для заполнения.

Шаблоны заполняются с использованием инструментов Jinja. Jinja позволяет использовать, как циклы и проверки в html шаблоне, так и методы объектов python.

Администраторский интерфейс создан с использование flask\_admin. Библиотека позволяет использовать созданные шаблоны по работе с бд. Также имеется возможность полностью вручную настроить отображение данные в админке. Достаточно создать класс, унаследованный от ModelView из flask\_admin.contrib.sqla. flask\_admin имеет множество различных параметров, изменяя которые можно получить свой собственный, ни на что не похожий, интерфейс.

Хочется также уделить внимание и реализации алгоритма авто очищения бд от устаревших данных о сессиях и связанных с ними местами. Для реализации подобного алгоритма была использована библиотека apscheduler. В данной библиотеке имеются различные типы запуска таймера. Тут следует остановить поподробнее. При локальном запуске нашего приложения активируется apscheduler.schedulers.background, который позволяет выполнять несколько процессов одновременно, не блокируя выполнение основного кода. Однако для удаленного сервера на Heroku требуется apscheduler.schedulers.blocking. Именно из-за этого в корне проекта лежат два файла, выполняющих одинаковую работу, но делающие ее по разным алгоритмам.

Сам процесс удаления не представляет собой чего-то сверхъественного. Каждые 24 часа (именно это число установлено в качестве интервала для таймера) запускается код, который получает из базы данные, которые были добавлены 1 неделю назад, и удаляет их. Тем самым происходит автоматическое самоочищение базы.

Таблицы описания классов приведены в приложении А.

**3 Описание программы для пользователя**

Проект представляет собой многостраничный сайт, в котором пользователь может ознакомиться с доступными фильмами и выбрать интересующий его фильм или подобрать фильм по нескольким критериям (год, жанр, продолжительность, режиссер). Выбрав фильм, пользователь может узнать краткое описание и жанр, длительность, год премьеры, а также актеров, принимавших участие в фильме, и режиссера. Кроме этого пользователь может ознакомиться с несколькими кадрами из фильма и трейлером к фильму. Ознакомившись, пользователь может забронировать билет. Для этого необходимо выбрать удобную дату, доступное время на сеанс и место в зале. Перед подтверждением брони пользователю необходимо ввести адрес электронной почты, на которую будут высланы билеты. Далее пользователь может, как заказать билеты на другое время или дату, так и вернуться к списку всех доступных фильмов в прокате.

На данный момент ознакомиться с проектом можно по ссылке:

<http://film-center-prj.herokuapp.com/>

После перехода по ссылке откроется главная страница сайта. Изображение главной страницы представлено на рисунке 3.1.

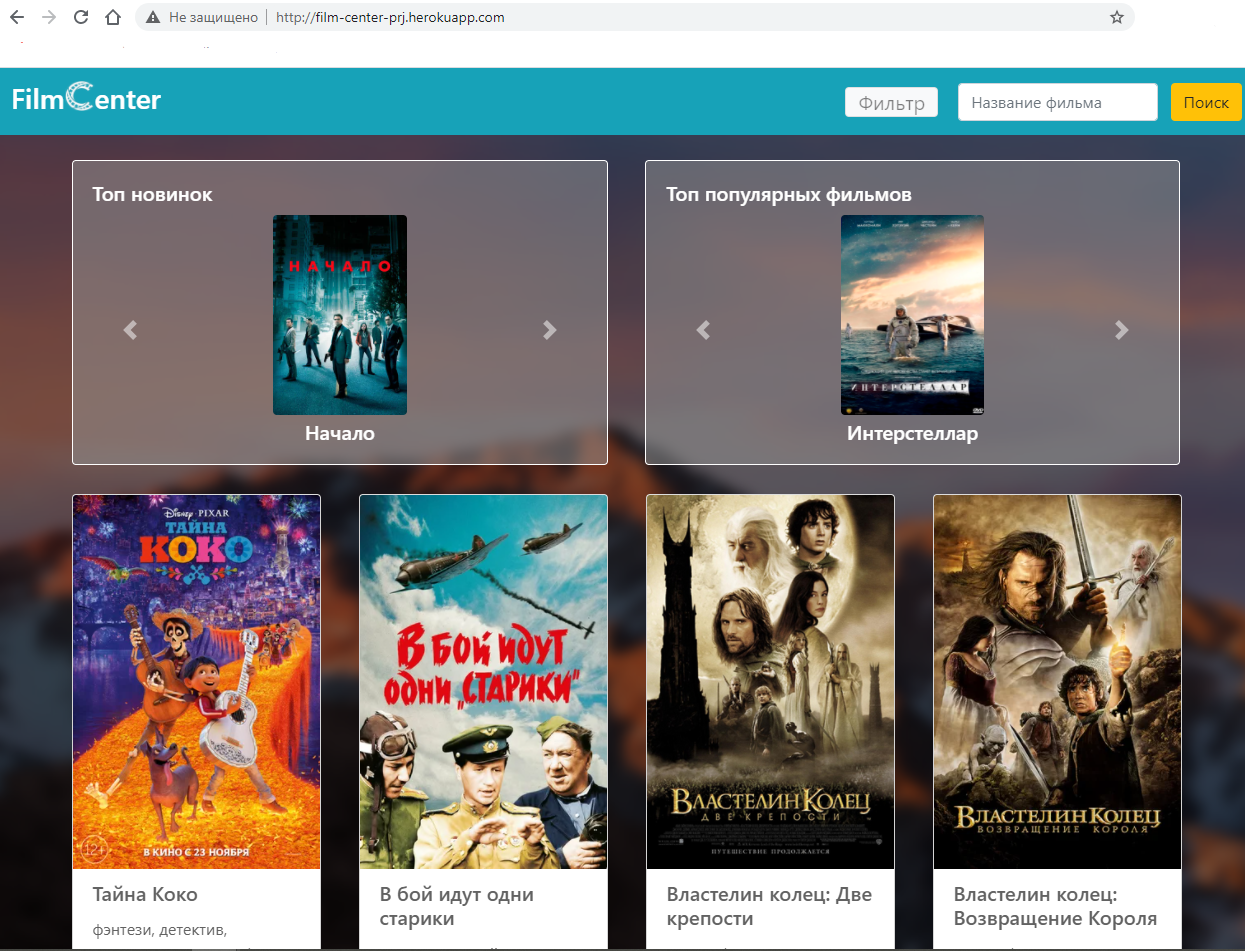


Рисунок 3.1 – Изображение главной страницы сайта.

В верхней части сайта находится панель с кнопкой открытия фильтра и поисковой строкой. Изображение панели представлено на рисунке 3.2.

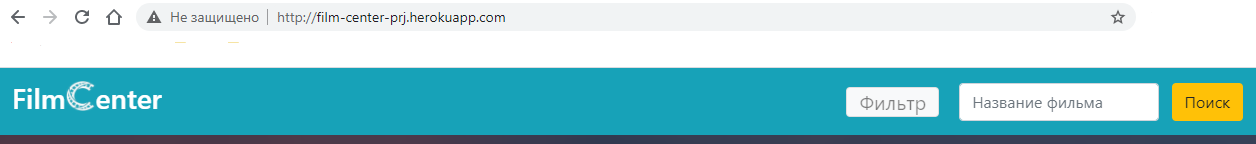


Рисунок 3.2 — Панель.

Чтобы вернуться на главную страницу необходимо нажать на ссылку «FilmCenter». Изображение ссылки представлено на рисунке 3.2.0.



Рисунок 3.2.0 – Изображение ссылки, ведущей на главную страницу.

Чтобы настроить фильтр нужно нажать кнопку «Фильтр». После появится блок с настройками поиска.

Чтобы выбрать, по какому параметру искать фильм необходимо выбрать нужный чекбокс и поставить в нем галочку, а также выбрать нужный элемент из выпадающего списка. Изображение окна фильтра представлено на рисунке 3.2.1.

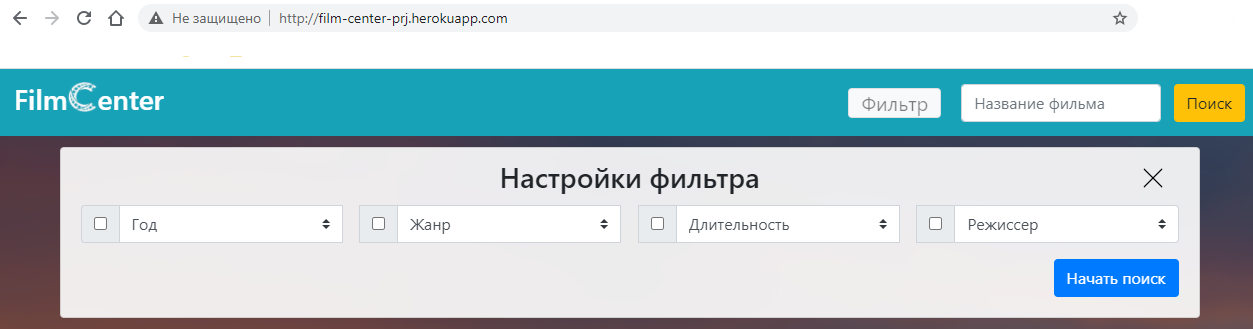


Рисунок 3.2.1. — Изображение фильтра

Чтобы перейти к более подробному описанию фильма необходимо нажать на карточку фильма. Изображение примера карточки представлено на рисунке 3.3.1. Изображение страницы с более подробным описанием фильма представлено на рисунке 3.3.2.



Рисунок 3.3.1. — Изображение примера карточки

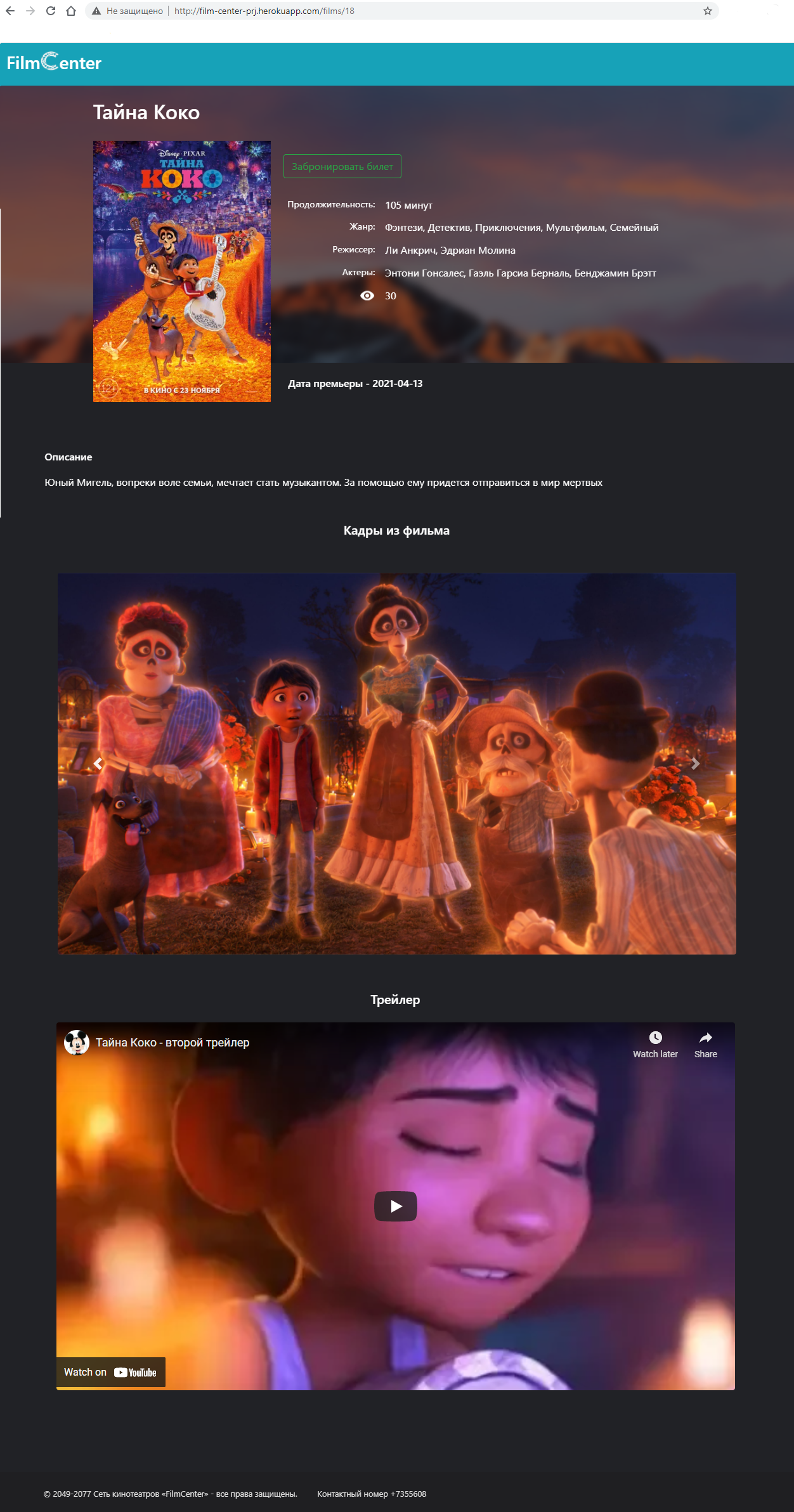


Рисунок 3.3.2. — Изображение страницы с более подробным описанием фильма.

Для бронирования билета необходимо нажать кнопку «Забронировать билет». В открывшейся странице необходимо выбрать дату и нажать на доступное время, чтобы продолжить. Изображение страницы с расписаниями представлено на рисунке 3.4.

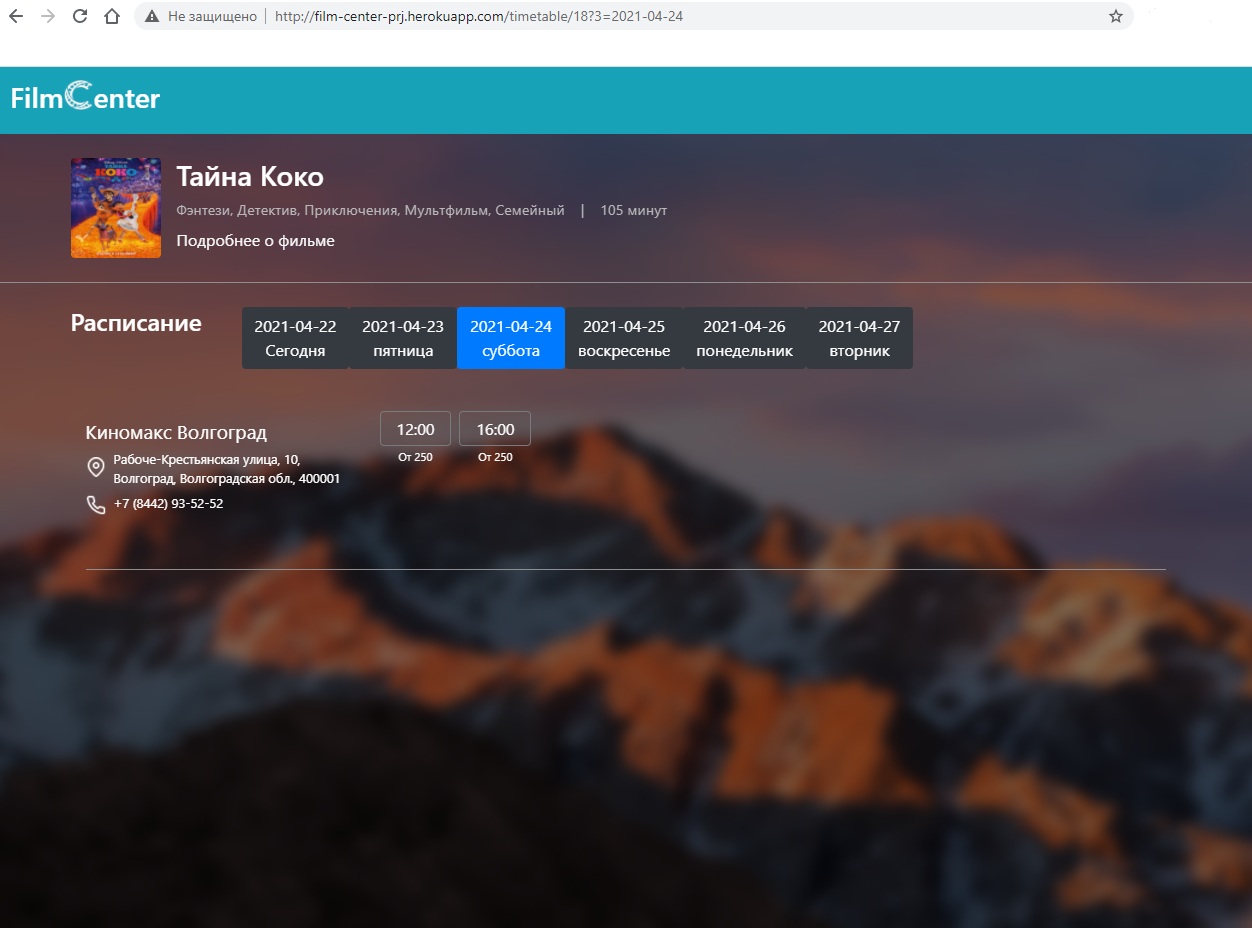


Рисунок 3.4 – Изображение страницы с расписаниями.

После выбора места откроется страница с выбором мест в зале. Можно выбрать неограниченное количество мест. И чтобы продолжить необходимо нажать кнопку «Продолжить». Изображение страницы с выбором мест представлено на рисунке 3.5.

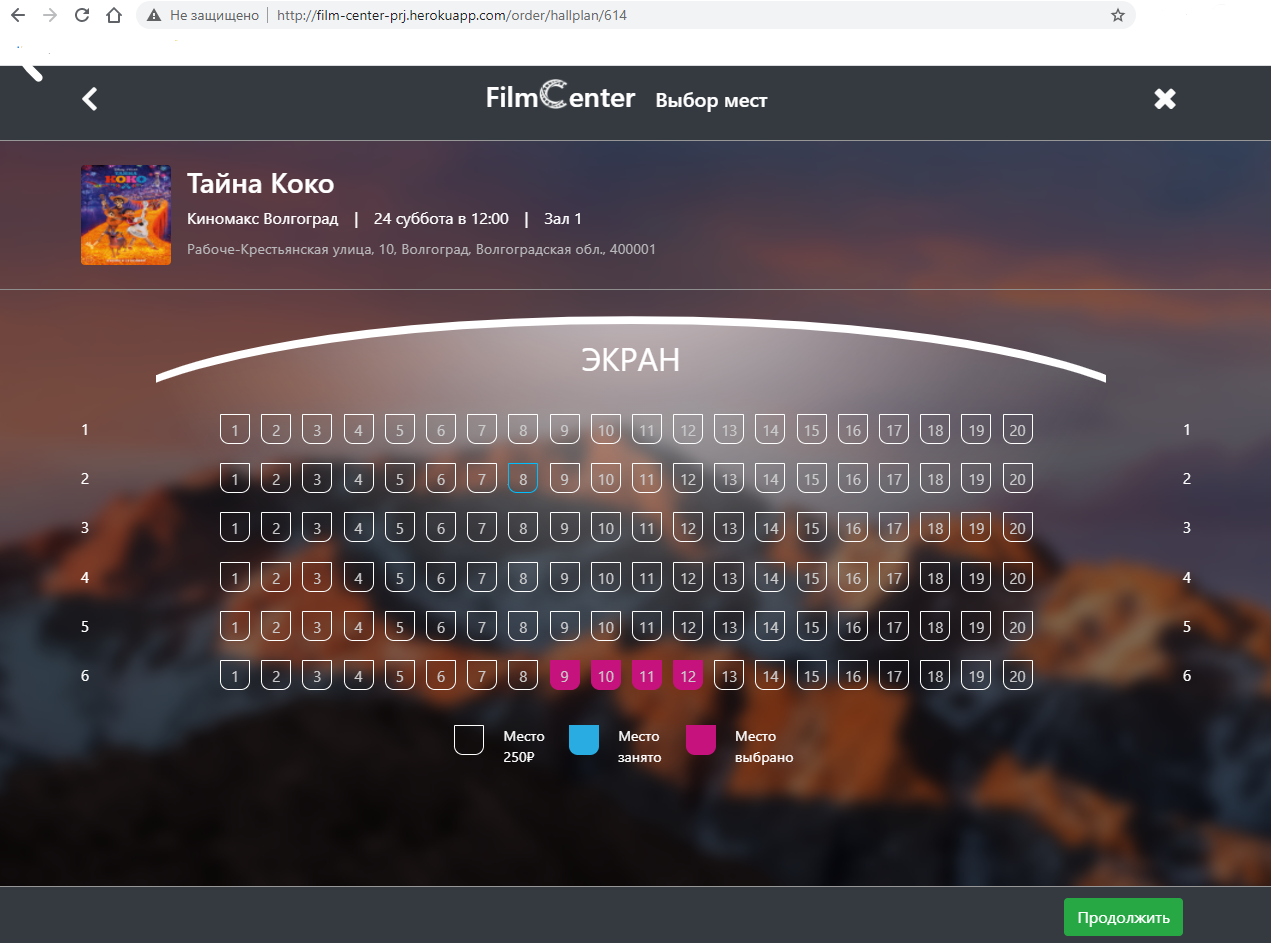


Рисунок 3.5 – Изображение страницы с выбором мест.

После нажатия кнопки «Продолжить» открывается окно с полем для ввода почты и галочкой подтверждения бронирования. Также в окне можно наблюдать выбранные билеты. Если необходимо удалить какой-либо билет, то нужно нажать на его крестик. Если удалены все билеты, то снова откроется окно с выбором мест в зале. Изображение страницы с подтверждением брони показано на рисунке 3.6.

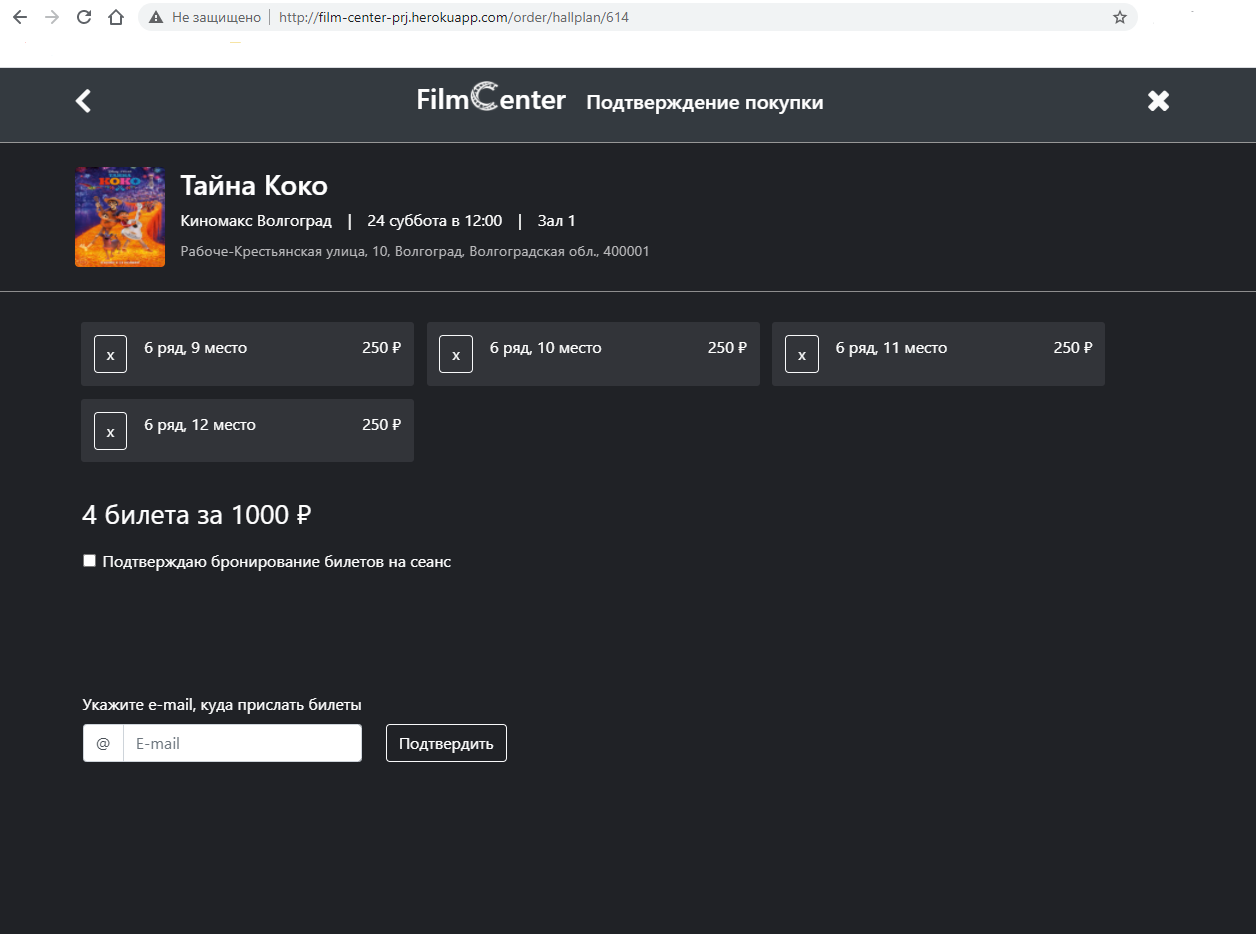


Рисунок 3.6 – Изображение страницы с подтверждением брони.

После ввода всех данных и нажатия кнопки «Подтвердить», откроется страница с расписанием и всплывет уведомление в правом верхнем углу, подсказывающее пользователю, что билеты были отправлены на почту. Изображения страницы с всплывающим уведомлением представлено на рисунке 3.7.

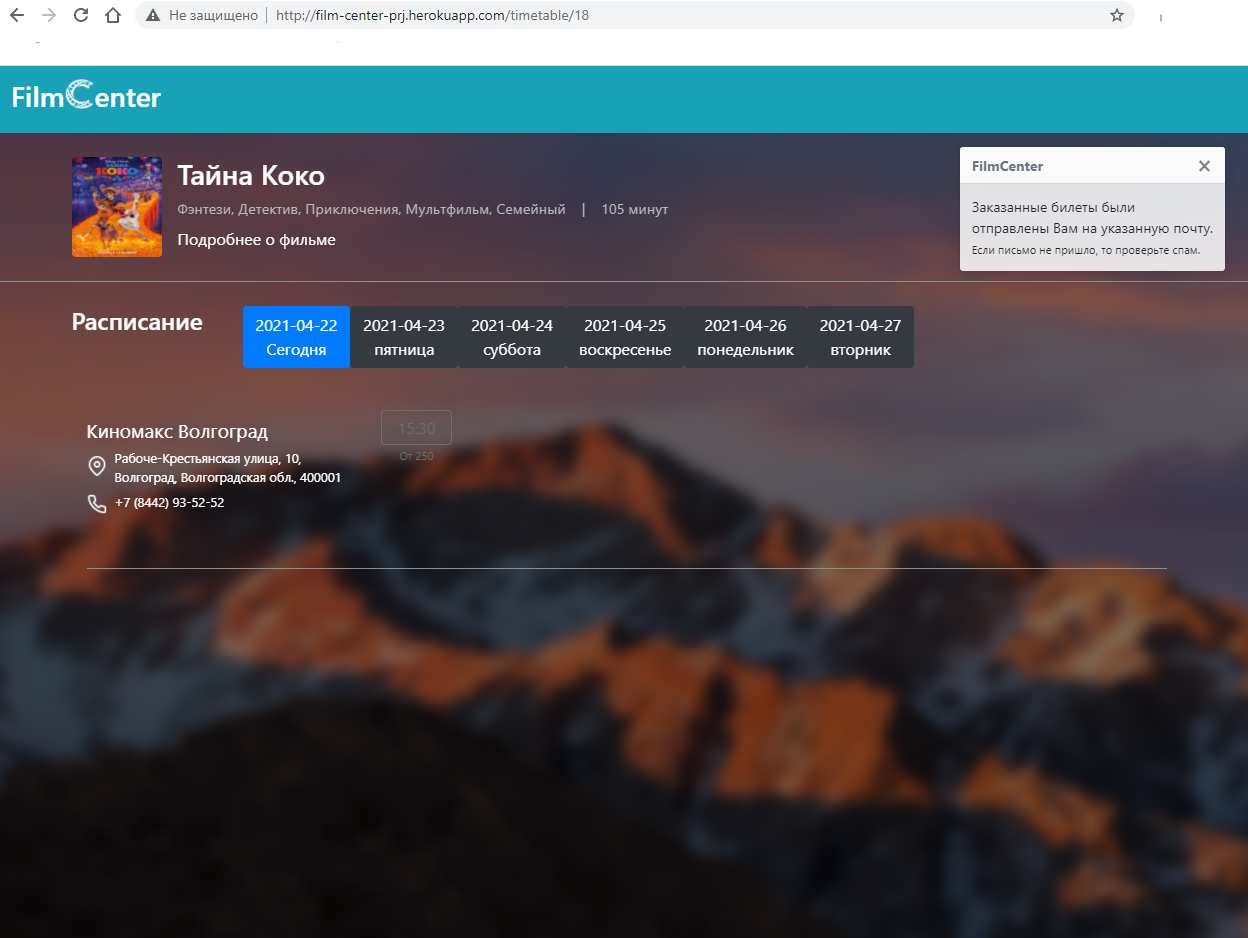


Рисунок 3.7 – Изображение страницы с всплывающим уведомлением.

Теперь проверим почту, и увидим, что пришло письмо с картинками заказанных билетов. Изображение письма представлено на рисунке 3.8.

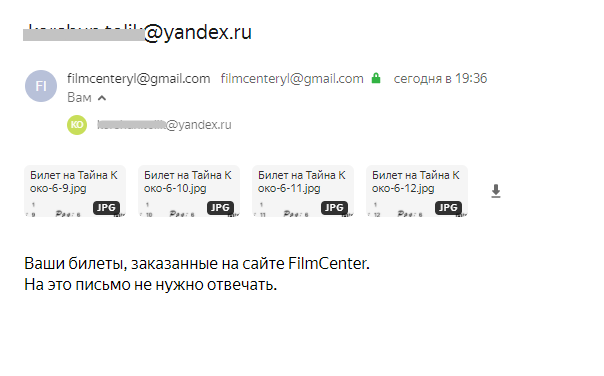


Рисунок 3.8 – Изображение письма от приложения.

Изображение билета представлено на рисунке 3.9. Билет содержит qrcode с уникальным идентификационным кодом, подтверждающий заказ билета.



Рисунок 3.9 – Изображение билета.

**4 Заключение**

Цель и задачи, поставленные в работе, выполнены. Создан сайт «FilmCenter», имеющий интуитивно-понятный графический интерфейс пользователя. Для реализации была выбрана среда разработки Pycharm, библиотеки Flask, Sqlalchemy, flask\_restful, flask\_admin и др. Программа предоставляет возможность ознакомиться с множеством фильмов, а также купить билет на доступный сеанс.

Сайт залит на сервис Heroku и работает в автономном режиме по следующему адресу: <http://film-center-prj.herokuapp.com/>



**Список использованных источников**

<https://docs.sqlalchemy.org/en/13/index.html>

<https://flask.palletsprojects.com/en/1.1.x/>

<https://jinja.palletsprojects.com/en/2.11.x/>

<https://www.heroku.com/>

**Приложение А**(справочное)  
Таблицы описания классов

Описание класса «FilmResource» представлено в таблице А.1.  
Таблица А.1 — описание класса «FilmResource».

|  |  |
| --- | --- |
| Метод | Описание |
| get | Возвращает найденный фильм по переданному id в формате json |
| delete | Удаляет найденный фильм по переданному id. |
| put | Изменяет найденный фильм по переданному id и аргументам |

Описание класса «FilmListResource» представлено в таблице А.2.  
Таблица А.2 — описание класса «FilmListResource».

|  |  |
| --- | --- |
| Метод | Описание |
| get | Возвращает список всех доступных фильмов из базы |
| post | Добавляет в базу фильм по переданным аргументам |

Описание класса «FilmSessionResource» представлено в таблице А.3.  
Таблица А.3 — описание класса «FilmSessionResource».

|  |  |
| --- | --- |
| Метод | Описание |
| get | Возвращает найденную сессию по переданному id в формате json |
| delete | Удаляет найденную сессию по переданному id. |
| put | Изменяет найденную сессию по переданному id и аргументам |

Описание класса «FilmSessionListResource» представлено в таблице А.4.  
Таблица А.4 — описание класса «FilmSessionListResource».

|  |  |
| --- | --- |
| Метод | Описание |
| get | Возвращает список всех доступных сессий из базы |
| post | Добавляет в базу сессию по переданным аргументам |

Описание класса «Genre» представлено в таблице А.5.  
Таблица А.5 — описание класса «Genre».

|  |  |
| --- | --- |
| Метод | Описание |
| \_\_repr\_\_ | Возвращает название жанра. |

Описание класса «FilmSession» представлено в таблице А.6.  
Таблица А.6 — описание класса «FilmSession».

|  |  |
| --- | --- |
| Метод | Описание |
| \_\_repr\_\_ | Возвращает строковое представление сессии на фильм. |

Описание класса «Film» представлено в таблице А.7.  
Таблица А.7 — описание класса «Film».

|  |  |
| --- | --- |
| Метод | Описание |
| \_\_repr\_\_ | Возвращает строковое представление фильма. |

Описание класса «Image» представлено в таблице А.8.  
Таблица А.8 — описание класса «Image».

|  |  |
| --- | --- |
| Метод | Описание |
| \_\_repr\_\_ | Возвращает строковое представление картинки к фильму. |

Описание класса «Place» представлено в таблице А.9.  
Таблица А.9 — описание класса «Place».

|  |  |
| --- | --- |
| Метод | Описание |
| \_\_repr\_\_ | Возвращает строковое представление забронированного места. |

Описание класса «Ticket» представлено в таблице А.10.  
Таблица А.10— описание класса «Ticket».

|  |  |
| --- | --- |
| Метод | Описание |
| \_\_init\_\_ | Загрузка всех данных. Загрузка шаблона билета. Вызов функции отрисовки qr кода. Вызов функции преобразования картинки в строку байт |
| draw\_text | Отрисовка всех переданных данных |
| make\_qrcode | Создание qr кода. |
| save\_tct | Преобразование картинки в строку |

Описание класса «AdminMixin» представлено в таблице А.11.  
Таблица А.11— описание класса «AdminMixin».

|  |  |
| --- | --- |
| Метод | Описание |
| is\_accessible | Возвращает True если администратор авторизирован. |
| inaccessible\_callback | Пересылает пользователя на страничку /login, если администратор не авторизирован. |