**Техническое задание**

**Проект «Что посмотреть»**

«Что посмотреть» — онлайн кинотеатр нового поколения. Смотрите новинки в лучшем качестве. Оставляйте отзывы, ставьте оценки и выбирайте только лучшее из мира большого кино.

**[1.](https://up.htmlacademy.ru/nodejs-api/4/project/what-to-watch" \l "task) Задача**

1.1. Разработать сервис для проекта «Что посмотреть».

1.2. Сервис предоставляет REST API интерфейс и CLI интерфейс.

1.3. Соединить бэкенд (сервис) с фронтендом (**дополнительное задание**).

**[2.](https://up.htmlacademy.ru/nodejs-api/4/project/what-to-watch" \l "scripts) Сценарии**

2.1. Добавление новой карточки с фильмом.

2.2. Редактирование карточки фильма.

2.3. Удаление карточки фильма.

2.4. Получение списка фильмов.

2.5. Получение списка фильмов определённого жанра.

2.6. Получение детальной информации по фильму.

2.7. Получение промо-фильма.

2.8. Получить список фильмов «К просмотру».

2.9. Добавить/удалить фильм из списка «К просмотру».

2.10. Получить список комментариев для определённого фильма.

2.11. Добавить новый комментарий к определённому фильму.

2.12. Создание новое пользователя.

2.13. Вход в закрытую часть приложения.

2.14. Выход из закрытой части приложения.

2.15. Проверка состояния пользователя.

**[3.](https://up.htmlacademy.ru/nodejs-api/4/project/what-to-watch" \l "entity) Сущности**

**[3.1.](https://up.htmlacademy.ru/nodejs-api/4/project/what-to-watch" \l "user-entity) Пользователь**

3.1.1. Пользователь представлен набором информации:

* Имя. Обязательное. Мин. длина 1 символ, макс. длина 15 символов.
* Электронная почта. Обязательное. Валидный адрес электронной почты.
* Аватар пользователя. Необязательное. Изображение пользователя в формате .jpg или .png.
* Пароль. Обязательное. Мин. длина 6 символов, макс. длина 12 символов.

3.1.2. В приложении не может быть двух пользователей с одинаковым email.

**[3.2.](https://up.htmlacademy.ru/nodejs-api/4/project/what-to-watch" \l "film-entity) Фильм**

3.2.1. Фильм представлен набором информации:

* Название фильма. Обязательное. Мин. длина 2 символа, макс. длина 100 символов.
* Описание фильма. Обязательное. Мин. длина 20 символов, макс. длина 1024 символа.
* Дата публикации фильма. Обязательное. По умолчанию — текущая дата.
* Жанр. Обязательное. Список доступных жанров предопределён: comedy, crime, documentary, drama, horror, family, romance, scifi, thriller.
* Год выхода. Обязательное;
* Рейтинг. Обязательное. Рассчитывается автоматически на основе оценок пользователей (средний бал). Значение по умолчанию — 0 (после создания фильма).
* Превью видео. Обязательное. Ссылка на превью видео.
* Видео. Обязательное. Ссылка на видео.
* Актёры. Обязательное. Список актёров, участвующих в фильме.
* Режиссёр. Обязательное. Мин. длина 2 символа, макс длина 50.
* Продолжительность фильма (в минутах). Обязательное.
* Количество комментариев. Рассчитывается автоматически.
* [Пользователь](https://up.htmlacademy.ru/nodejs-api/4/project/what-to-watch" \l "user-entity). Обязательное. Ссылка на сущность «Пользователь».
* Постер. Обязательное. Ссылка на постер для фильма в формате .jpg.
* Фоновое изображение. Обязательное. Ссылка на изображение для фона в формате .jpg.
* Цвет фона. Обязательное.

**[3.3.](https://up.htmlacademy.ru/nodejs-api/4/project/what-to-watch" \l "comment-entity) Комментарий**

3.3.1. Комментарий представлен набором информации:

* Текст комментария. Обязательное. Мин. длина 5 символов, макс. длина 1024 символа.
* Рейтинг. Пользовательская оценка. Обязательное. Значение от 1 до 10.
* Дата публикации комментария.
* [Автор комментария](https://up.htmlacademy.ru/nodejs-api/4/project/what-to-watch" \l "user-entity). Обязательное. Ссылка на сущность «Пользователь».

**[4.](https://up.htmlacademy.ru/nodejs-api/4/project/what-to-watch" \l "func) Функциональность**

**[4.1.](https://up.htmlacademy.ru/nodejs-api/4/project/what-to-watch" \l "cli) Command Line Interface**

4.1.1. Приложение предоставляет Command Line Interface (CLI). Для запуска CLI предусмотрен отдельный сценарий в package.json — cli.

4.1.2. Модуль, отвечающий за запуск CLI, содержит корректный shebang.

4.1.3. CLI поддерживает обработку аргументов:

* --version. Выводит информацию о версии приложения. Информация о версии считывается из package.json. Пример вывода: 1.0.1.
* --help. Выводит список и описание всех поддерживаемых аргументов. Оформление списка остаётся на усмотрение студента. Пример оформления продемонстрирован ниже.

Программа для подготовки данных для REST API сервера.

Пример: cli.js --<command> [--arguments]

Команды:

--version: # выводит номер версии

--help: # печатает этот текст

--import <path>: # импортирует данные из TSV

--generate <n> <path> <url> # генерирует произвольное количество тестовых данных

* --generate <n> <filepath> <url>. Создаёт файл в формате tsv с тестовыми данными. Параметр n задаёт количество генерируемых карточек для фильмов. Параметр filepath указывает путь для сохранения файла с описанием карточек фильмов. Параметр <url> задаёт адрес сервера, с которого необходимо взять данные. Каждая строка в файле tsv содержит всю необходимую информацию для создания карточки фильма и пользователя. Информация о фильме включает сведения о пользователе. Список полей для карточки фильма и пользователя представлен в разделе с описанием сущностей.
* --import <filepath>. Импортирует в базу данных информацию из tsv-файла. Путь к файлу передаётся в параметре filepath.

4.1.4. Запуск CLI без аргументов приводит к исполнению команды --help.

**[4.2.](https://up.htmlacademy.ru/nodejs-api/4/project/what-to-watch" \l "rest) REST API в формате OpenAPI**

4.2.1. Приложение предоставляет REST API интерфейс. Имена ресурсов, маршрутов остаются на усмотрение студента.

4.2.2. Спецификация REST API представлена в формате OpenAPI.

4.2.3. Для описания спецификации используется язык YAML.

4.2.4. Спецификация сохранена в директории specification (в корне проекта).

**[4.3.](https://up.htmlacademy.ru/nodejs-api/4/project/what-to-watch" \l "service) REST API сервис**

4.3.1. Сервис предоставляет ресурсы в соответствии со [спецификацией](https://up.htmlacademy.ru/nodejs-api/4/project/what-to-watch" \l "rest).

4.3.2. Настройки, необходимые для запуска и работы сервиса, передаются с помощью переменных окружения.

4.3.3. Список всех переменных окружения приведён в файле Workflow.md, который расположен в корне проекта. Формат описания: PARAM=value — описание.

4.3.4. Для удобства передачи параметров окружения используются .env-файлы. Они не должны сохраняться в системе контроля версий.

4.3.5. В корне проекта расположен .env.example. Он содержит список всех переменных окружения, необходимых для работы приложения. Переменным заданы значения по умолчанию (для примера использования).

**[4.4.](https://up.htmlacademy.ru/nodejs-api/4/project/what-to-watch" \l "service-start) Запуск сервиса**

4.4.1. При старте сервис проверяет список установленных переменных окружения. Если какая-то переменная не задана, сервис выдаёт сообщение об ошибке и прекращает выполнение.

4.4.2. В файле Workflow.md предусмотрите раздел «Запуск проекта» и опишите в нём подробную инструкцию по запуску проекта.

4.4.3. В файле Workflow.md в секции «Сценарии» перечислите все сценарии, доступные в package.json и опишите их предназначение.

В корневой директории проекта расположен файл docker-compose.yml для быстрого разворачивания базы данных MongoDB с помощью Docker (*если вы используете в работе docker*).

**[5.](https://up.htmlacademy.ru/nodejs-api/4/project/what-to-watch" \l "resource) Список ресурсов**

**[5.1.](https://up.htmlacademy.ru/nodejs-api/4/project/what-to-watch" \l "add-film) Добавление нового фильма**

5.1.1. Сервис предоставляет ресурс для добавления новых фильмов.

5.1.2. Добавлять новые фильмы могут только авторизованные клиенты.

5.1.3. Для добавления нового фильма клиент передаёт информацию, указанную в пункте 3.2, за исключением значений, которые рассчитываются автоматически.

5.1.4. Перед добавлением фильма данные, полученные от клиента, валидируются. Если валидация не пройдена, сервис возвращает ошибку и корректный код состояния. Фильм при этом не создаётся.

5.1.5. В случае успешного добавления нового фильма сервис в ответе возвращает информацию о созданном фильме (объект фильма).

**[5.2.](https://up.htmlacademy.ru/nodejs-api/4/project/what-to-watch" \l "edit-film) Редактирование карточки фильма**

5.2.1. Сервис предоставляет ресурс для редактирования карточки фильма.

5.2.2. Клиент может редактировать карточки фильмов, которые создал он.

5.2.3. Для редактирования карточки фильма клиент передаёт обновлённую информацию по фильму, указанную в пункте 3.2 (за исключением значений, которые рассчитываются автоматически), и идентификатор редактируемого фильма.

5.2.4. Полученные от клиента данные проходят валидацию. Если валидация не пройдена сервис возвращает информацию об ошибке и корректный код состояния.

5.2.5. В случае успешного обновления карточки фильма сервис возвращает клиенту обновлённый объект фильма.

5.2.6. Редактировать карточки фильмов могут только авторизованные клиенты.

**[5.3.](https://up.htmlacademy.ru/nodejs-api/4/project/what-to-watch" \l "delete-film) Удаление карточки фильма**

5.3.1. Сервис предоставляет ресурс для удаления фильмов. Клиент может удалять только «свои» фильмы.

5.3.2. При попытке клиентом удалить фильм другого пользователя сервис возвращает ошибку и корректный код состояния.

5.3.3. При удалении фильма комментарии к нему удаляются автоматически.

5.3.4. Удалять фильмы могут только авторизованные клиенты.

**[5.4.](https://up.htmlacademy.ru/nodejs-api/4/project/what-to-watch" \l "get-films) Получение списка фильмов**

5.4.1. Сервис предоставляет ресурс для получения списка фильмов. По умолчанию сервис возвращает не больше 60 фильмов.

5.4.2. Клиент может запросить больше фильмов за раз, указав нужное количество в параметре запроса.

5.4.3. Сервис всегда возвращает список фильмов, отсортированный по дате публикации (по убыванию).

5.4.6. Список возвращаемых полей для фильма: название фильма, дата публикации, жанр, превью видео, пользователь (который добавил фильм), постер фильма, количество комментариев.

5.4.7. Список могут запрашивать авторизованные и анонимные клиенты.

**[5.5.](https://up.htmlacademy.ru/nodejs-api/4/project/what-to-watch" \l "get-films-by-genre) Получение списка фильмов определённого жанра**

5.5.1. Сервис предоставляет ресурс для получения списка фильмов определённого жанра.

5.5.2. Требования к ресурсы аналогичны ресурсу «Получение списка фильмов».

**[5.6.](https://up.htmlacademy.ru/nodejs-api/4/project/what-to-watch" \l "get-film-details) Получение детальной информации по фильму**

5.6.1. Сервис предоставляет ресурс для получения детальной информации по одному фильму.

5.6.2. Информация по фильму содержит все поля, перечисленные в пункте [3.2](https://up.htmlacademy.ru/nodejs-api/4/project/what-to-watch" \l "film-entity).

5.6.3. Детальная информация по фильму включает сведения о количестве комментариев, но сами комментарии не включены.

5.6.4. Детальную информацию по фильму могут запрашивать авторизованные и анонимные клиенты.

**[5.7.](https://up.htmlacademy.ru/nodejs-api/4/project/what-to-watch" \l "get-promo-film) Получение промо-фильма**

5.7.1. Сервис предоставляет ресурс для получения информации о промо-фильме.

5.7.2. Информацию о промо-фильме могут запрашивать анонимные и авторизованные клиенты.

5.7.3. Информация о промо-фильме содержит все поля, перечисленные в разделе [3.2](https://up.htmlacademy.ru/nodejs-api/4/project/what-to-watch" \l "film-entity).

**[5.8.](https://up.htmlacademy.ru/nodejs-api/4/project/what-to-watch" \l "get-films-to-watch) Получить список фильмов «К просмотру»**

5.8.1. Сервис предоставляет ресурс для получения списка фильмов, добавленных к просмотру.

5.8.2. Сервис возвращает все фильмы, добавленные к просмотру.

5.8.3. Список возвращаемых полей для фильма соответствует пункту 5.4.6.

5.8.4. Ресурс доступен только авторизованным клиентам.

**[5.9.](https://up.htmlacademy.ru/nodejs-api/4/project/what-to-watch" \l "add-delete-to-watch) Добавить/удалить фильм из списка «К просмотру»**

5.9.1. Сервис предоставляет ресурс для добавления и удаления любого фильма в/из списка к просмотру.

5.9.2. Ресурс доступен только авторизованным клиентам.

**[5.10.](https://up.htmlacademy.ru/nodejs-api/4/project/what-to-watch" \l "get-film-comments) Получить список комментариев для определённого фильма**

5.10.1. Сервис предоставляет ресурс для получения списка комментариев для определённого фильма.

5.10.2. Сервис всегда возвращает список из 50 (или меньше) последних комментариев.

5.10.3. Комментарии отсортированы по дате публикации (по убыванию). Сначала новые, потом старые.

5.10.4. Комментарий в списке представлен набором информации перечисленной в пункте [3.3](https://up.htmlacademy.ru/nodejs-api/4/project/what-to-watch" \l "comment-entity).

5.10.5. Комментарии могут запрашивать авторизованные и анонимные клиенты.

**[5.11.](https://up.htmlacademy.ru/nodejs-api/4/project/what-to-watch" \l "add-comment) Добавить новый комментарий к определённому фильму**

5.11.1. Сервис предоставляет ресурс для добавления комментариев к определённому фильму.

5.11.2. Добавлять комментарии могут только авторизованные клиенты.

5.11.3. Для добавления нового комментария для фильма клиент передаёт набор информации, указанный в пункте [3.3](https://up.htmlacademy.ru/nodejs-api/4/project/what-to-watch" \l "comment-entity), и идентификатор фильма, для которого создаётся комментарий.

5.11.4. Перед добавлением комментария сервис валидирует данные. В случае ошибки валидации сервис возвращает ошибку и корректный код состояния.

5.11.5. В случае успешного добавления комментария сервис возвращает информацию о созданном комментарии (объект комментария).

5.11.6. Удаление и редактирование комментариев не предусмотрено.

5.11.7. Добавление комментария приводит к обновлению рейтинга фильма.

**[5.12](https://up.htmlacademy.ru/nodejs-api/4/project/what-to-watch" \l "create-user) Создание новое пользователя**

5.12.1. Сервис предоставляет ресурс для создания нового пользователя.

5.12.2. Для создания нового пользователя клиент отправляет запрос и передаёт данные, указанные в пункте [3.1](https://up.htmlacademy.ru/nodejs-api/4/project/what-to-watch" \l "user-entity).

5.12.3. Создавать новых пользователей могут только анонимные клиенты.

5.12.4. Email создаваемого пользователя уникальный. В системе не может быть двух пользователей с одинаковым email.

5.12.5. Данные, полученные от клиента, проходят валидацию. Если проверка не пройдена, сервер возвращает ошибку и корректный код состояния.

5.12.6. В случае успешного создания пользователя сервис возвращает информацию о созданном пользователе (объект пользователя). Возвращаемая информация не содержит конфиденциальных данных.

5.12.7. Пароль пользователя не хранится в базе данных в открытом виде. Вместо этого хранится хеш пароля.

**[5.13](https://up.htmlacademy.ru/nodejs-api/4/project/what-to-watch" \l "user-login) Вход в закрытую часть приложения**

5.13.1. Сервис предоставляет ресурс для аутентификации и авторизации пользователя.

5.13.2. Для входа в закрытую часть приложения клиент передаёт логин (email) и пароль пользователя.

5.13.3. Если логин или пароль неверный или пользователь не существует, сервер возвращает ошибку и соответствующий код состояния.

5.13.4. В случае успешной проверки логина и пароля сервис возвращает клиенту токен.

5.13.5. Клиент использует этот токен для отправки любых запросов. Токен передаётся в заголовках.

**[5.14](https://up.htmlacademy.ru/nodejs-api/4/project/what-to-watch" \l "user-logout) Выход из закрытой части приложения (только если авторизация выполняется на основе сессий)**

5.14.1. Сервис предоставляет ресурс для завершения пользовательского сеанса.

5.14.2. Ресурс доступен только авторизованным клиентам.

5.14.3. Если токен прошёл проверку, сервис завершает пользовательский сеанс (сессию). В противном случае возвращает ошибку и соответствующий код состояния.

**[5.15](https://up.htmlacademy.ru/nodejs-api/4/project/what-to-watch" \l "user-check) Проверка состояния пользователя**

5.15.1. Сервис предоставляет ресурс для проверки состояния клиента — авторизован или нет.

5.15.2. Проверка состояния клиента производится на основании сессионного токена.

5.15.3. Если клиент авторизован, сервис возвращает соответствующий код состояния и информацию о пользователе (смотри пункт [3.1](https://up.htmlacademy.ru/nodejs-api/4/project/what-to-watch" \l "user-entity)). Информация о пользователе не должна содержать конфиденциальные данные (пароль).

**[6.](https://up.htmlacademy.ru/nodejs-api/4/project/what-to-watch" \l "validation) Валидация**

6.1. Данные, полученные от клиента, проверяются перед использованием (приводятся к нужному типу, проверяются идентификаторы для связанных сущностей и так далее).

6.2. Данные, которые не удовлетворяют правилам валидации, не могут использоваться для создания или редактирования сущностей. В таких ситуациях сервис возвращает ошибку и соответствующий код состояния.

**[7.](https://up.htmlacademy.ru/nodejs-api/4/project/what-to-watch" \l "mistakes) Ошибки**

7.1. В случае невозможности выполнения запроса сервис возвращает клиенту список ошибок. Ответ с ошибкой содержит описание ошибки и соответствующий код состояния.

7.2. Формат описания ошибок и их детализация остаются на усмотрение студента.

**[8.](https://up.htmlacademy.ru/nodejs-api/4/project/what-to-watch" \l "frontend) Фронтенд (необязательное задание)**

8.1. Фронтенд для работы с сервисом предоставляется в виде готового приложения. Интеграция бэкенда и фронтенда является *дополнительным* заданием.

8.2. В readme.md к фронтенду приводится описание ресурсов, к которым обращается приложение.

8.3. Студент самостоятельно обновляет во фронтенде список маршрутов и при необходимости корректирует структуры данных.

8.4. Допускается внесение любых изменений во фронтенд.

8.5 Обязательные сценарии:

* Регистрация пользователя;
* Вход пользователя в закрытую часть приложения;
* Проверка состояния пользователя (проверка токена);
* Главная страница (список фильмов, переключение между жанрами);
* Добавление нового фильма;
* Просмотр карточки фильма;
* Добавление и просмотр комментариев.

**[9.](https://up.htmlacademy.ru/nodejs-api/4/project/what-to-watch" \l "requirements) Технические требования**

9.1. Проект запускается и работает в актуальной версии Node.js.

9.2. Для разработки проекта используется TypeScript.

9.3. Для хранения данных используется MongoDB.

9.4. Для обработки запросов от клиентов используется Express.js.

9.5. Для работы с большими файлами (или файлами, которые потенциально могут стать большими) используются потоки (Streams).

9.6. В коде не должно быть зашито данных, которые могут меняться при переносе в продуктовую среду. Например: доступы для подключения к СУБД, адрес сервера для получения тестовых данных и так далее. Все перечисленные данные должны устанавливаться через переменные окружения.

9.7. Замена пакетов, не перечисленных в технических требованиях — допускается по согласованию с наставником.