ФГБОУ ВО «ПЕТРОЗАВОДСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
ИНСТИТУТ МАТЕМАТИКИ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Отчет по дисциплине «Верификация программного обеспечения»

Выполнила:
студентка группы 22605
Панфилова О.С.
Научный руководитель:
к.ф-м.н., доцент
К. А. Кулаков

.

Петрозаводск 2019

Содержание

1 Объект тестирования

В рамках курса «Верификация программного обеспечения» будет рассмотрено приложение, обучающее основам Angular.

Вы можете щелкнуть две ссылки над панелью управления («Главная» и «Фильмы»), чтобы перемещаться между панелью управления и страницей со списком фильмов.

Если щелкнуть на фильм, маршрутизатор откроет представление «Сведения о фильме», в котором можно изменить наименование фильма.

На странице «Сведения о фильме», нажав кнопку «Назад», вы вернетесь на предыдущую страницу, на которой вы находились («Фильмы или «Панель управления». Если вы нажмете «Фильмы», приложение отобразит представление главного списка «Фильмы».

Приложение написано при помощи фреймворка Angular v.6.

Приложение состоит из следующих страниц:

1. Панель инструментов – на странице доступно текстовое поля для поиска фильма по его названию, две кнопки: «Главная» (при нажатии выполняется переход на панель инструментов) и «Фильмы» (при нажатии выполняется переход на страницу «Фильмы»).
2. Фильмы – страница, на которой можно добавить фильм и отображается список всех фильмов в произвольном порядке. Также здесь можно удалить фильм, нажав на иконку-крестик рядом с фильмом. При нажатии на наименование фильма выполняется переход на страницу «Детали фильма».
3. Детали фильма – на странице отображается информация о фильме (идентификатор записи и наименование фильма).

Все сообщения об ошибках должны выводится на странице браузера.

У фильма есть наименование. Имя представляет из себя текстовое значение с максимальной длиной 100 символов.

1.1 Функциональность объекта тестирования

В приложении реализованы следующие функции:

1. Создание фильма.
2. Удаление фильма.
3. Обновление информации о фильме.
4. Просмотр информации о фильме.
5. Просмотр списка фильмов.

В аттестационном тестировании принимают участие все вышеописанные функции.

1.2 Требования

- T1: Приложение должно работать в браузере Google Chrome версии 78.0.3904.108.
- T2: Требований к дизайну не предъявляется. Система может быть оформлена в любом цветовом решении.
- T3: На всех страницах системы должны быть ссылки, чтобы перемещаться между панелью управления и страницей со списком фильмов.
- T4: Если щелкнуть на фильм, должна открыться страница «Сведения о фильме».

- Т5: На странице «Сведения о фильме», нажав на кнопку «Назад», должно выполняться перемещение на предыдущую страницу, на которой находился пользователь.

1.3 Функциональные требования

- ФТ1: На странице «Сведения о фильме можно изменить информацию о фильме. Менять можно только наименование фильма.
- ФТ2: Должна быть возможность отображения списка фильмов.
- ФТ3: Должна быть возможность просмотра информации о фильме.
- ФТ4: Должно выполняться удаление фильма.
- ФТ5: Должна быть возможность добавить фильм (наименование). Максимальная длина наименования фильма: 100 символов. Минимальная: 1 символ. Тип поля: текстовое.

2 Стратегия тестирования

2.1 Описание структуры объекта тестирования и связей внутри объекта тестирования (архитектура)

Архитектура приложения соответствует компонентной архитектуре Angular приложения [1]. Больше информации о конфигурационных файлах, структуре проекта, исходных файлах приложения, тестовых файлы и др. можно получить в документации Angular, глава «Configuration», подраздел «Project File Structure», пункт «File structure»:

Связи между блоками описаны в документации Angular, глава «Architecture», подраздел «Architecture overview», пункт «What's next» [3]:

Фреймворк состоит из нескольких библиотек (или модулей), каждая из которых содержит в себе определенный функционал, а каждый модуль состоит из совокупности классов их свойств и методов.

Файлы на верхнем уровне папки src/ поддерживают тестирование и запуск приложения. Подпапки содержат исходный код приложения и конкретную конфигурацию приложения.

Подробнее о назначении файлов и каталогов внутри src/ описано в документации Angular: глава «Configuration», подраздел «Project File Structure», пункт «Application project files» [4].

Диаграмма архитектуры определяет восемь основных строительных блоков приложения Angular (рис.1):

1. Модули.
2. Компоненты.
3. Шаблоны.
4. Метаданные.
5. Привязка данных.
6. Директивы.
7. Сервисы.
8. Внедрение зависимости.

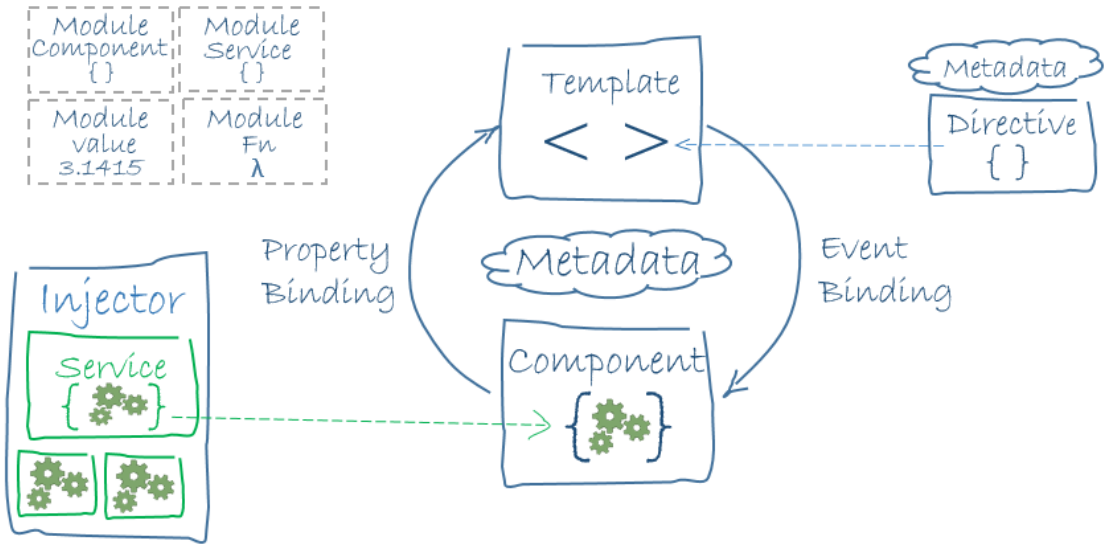


Рис.1: Архитектура приложений Angular.

. Опишем назначение конфигурационных блоков внутри папки src/ (рис.2).

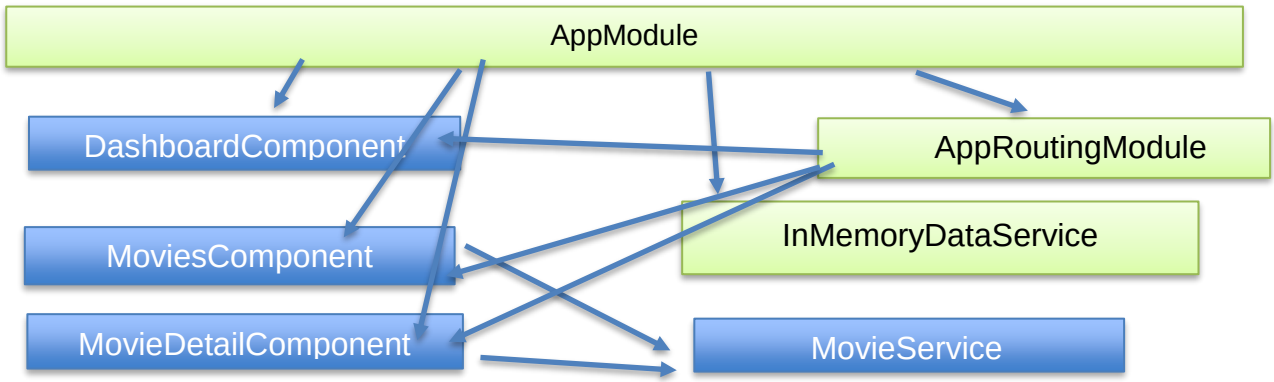


Рис.2: Конфигурационные блоки папки src/

В приложении есть один *корневой модуль AppModule*, а также AppRoutingModule, который содержит в себе настройки маршрутизации для всего приложения.

Компоненты:

1. DashboardComponent - управляет отображением панели инструментов на экране.

2. MoviesComponent - управляет отображением списка фильмов на экране (страница «Фильмы»).
3. MovieDetailComponent - управляет отображением страницы «Детали фильма».

Сервисы:

1. MoviesService – поставщик данных о фильме / фильмах.
2. InMemoryDataService – вспомогательный сервис для тестирования.

Все компоненты и корневой взаимодействуют с сервисом MovieService. Корневой модуль также взаимодействует с InMemoryDataService.

2.2 Отношение к тестированию структурных элементов

Тестируемые компоненты выделены голубым цветом, нетестируемые – зеленым (рис.2).

Тестирование будет проводиться для всех вышеописанных структурных элементов, кроме

1. InMemoryDataService, т.к. он используются для эмуляции работы сервера и базы данных.
2. AppRoutingModule, т.к. это файл настроек маршрутизации к компонентам. Стоит отметить, что AppRoutingModule не содержит функций и представляет из себя модуль с настройками маршрутизации корневого модуля для удобства вынесенного в отдельный файл, и тестироваться он не будет. Будет тестироваться только взаимодействие этого модуля с компонентами, т.е. успешная установка настроек маршрутизации между компонентами, которая производится с помощью специальных конструкций Angular.

3. AppModule – точка входа в приложение.

В таблице ниже подробнее описаны тестируемые сервисы и компоненты.

Сервис и его назначение	Тестируемая функция и ее назначение	Вид тестирования
MovieService – Назначение: поставщик данных о фильме / фильмах	getMovies (): Observable<Movie[]> Назначение: Получение списка фильмов. Возвращает список фильмов.	Блочное
	getMovie(id: number): Observable<Movie> Назначение: получение информации о фильме. Получает идентификатор фильма, который нужно получить с сервера. Возвращает фильм с сервера.	Блочное
	addMovie (newMovie: Movie): Observable<Movie> Назначение: добавление фильма. Получает свойства создаваемого фильма.	Блочное

	Возвращает с сервера созданный фильм.	
	deleteMovie (movie: Movie | number): Observable<Movie> Назначение: удаление фильма. Получает фильм, который нужно удалить. Возвращает удаленный фильм.	Блочное
	updateMovie (movie: Movie): Observable<any> Назначение: обновление информации о фильме. Получает фильм, который нужноотредактировать. Возвращает обновленную информацию о фильме с сервера.	Блочное
	constructor(private http: HttpClient, private messageService: MessageService) Назначение: специальный метод (функция) класса. Конструктор вызывается при создании	Блочное

	объекта класса. Определяет зависимости компонента.	
Компонент	Тестируемый метод	Тип тестирования
DashboardComponent Назначение: управляет отображением панели инструментов на экране.	ngOnInit() Назначение – вызывается один раз после установки свойств компонента, которые участвуют в привязке. Выполняет инициализацию компонента. Взаимодействует с MovieService	Интеграционное
	constructor(private movieService: MovieService) { } Назначение: специальный метод (функция) класса. Конструктор вызывается при создании объекта класса. Определяет зависимости компонента.	Блочное

MoviesComponent Назначение: управляет отображением списка фильмов на экране (страница «Фильмы»).	ngOnInit() Назначение – вызывается один раз после установки свойств компонента, которые участвуют в привязке. Выполняет инициализацию компонента. Взаимодействует с MovieService	Интеграционное
	add(name: string): void Назначение: добавление фильма. Взаимодействует с MovieService Получает новое имя фильма.	Интеграционное
	delete(movie: Movie): void Назначение: удаление фильма. Взаимодействует с MovieService. Получает информацию о фильме. Который нужно удалить.	Интеграционное
MovieDetailComponent	goBack() Назначение: возвращение на предыдущую страницу.	Блочное

Назначение: управляет отображением страницы «Детали фильма»	constructor(private movieService: MovieService) Назначение: специальный метод (функция) класса. Конструктор вызывается при создании объекта класса. Определяет зависимости компонента.	Блочное
	ngOnInit() Назначение – вызывается один раз после установки свойств компонента, которые участвуют в привязке. Выполняет инициализацию компонента Взаимодействует с MovieService	Интеграционное
	save() Назначение: обновление информации о фильме Взаимодействует с MovieService Назначение: обновление информации о фильме.	Интеграционное
	constructor (private route: ActivatedRoute,	Блочный

	private movieService: MovieService, private location: Location) Назначение: специальный метод (функция) класса. Конструктор вызывается при создании объекта класса. Определяет зависимости компонента.	

2.3 Основные положения процедуры проведения тестирования

1. Тест считается успешно пройденным, если ожидаемый и фактический результат теста совпадают.

2. Этап тестирования считается завершенным в том случае, когда 80 % тестов текущего этапа успешно пройдены.

3. Переход к следующему этапу тестирования может быть совершен только в том случае, если предыдущий этап тестирования завершен.

4. Процедура проведения тестирования может быть остановлена только в случае ошибки, блокирующей выполнение тестов.

5. Возобновление процедуры проведения тестирования происходит после исправления блокирующей ошибки, которая помешала предыдущему выполнению

2.4 Инструменты тестирования

Для проведения блочного и интеграционного тестирования будут использованы следующие инструменты.

1. **Karma** - инструмент для выполнения исходного кода с тестовым кодом в браузере. Будет использоваться для блочного тестирования.

2. **Protractor** - фреймворк комплексных тестов для Angular. Он запускает тесты в настоящем браузере и взаимодействует с обозревателем для тестирования логики приложения.

3. **Утилиты тестирования Angular (**Angular Testing Utilities) **-** набор классов и функций, необходимых для создания тестового окружения для кода Angular. Их можно найти в документации к Angular api глава «Dev workflow», раздел «Testing» [5].

2.5 Стратегия блочного тестирования

Первый вид тестирования, которому будет подвержена система, будет блочное тестирование.

Данный вид тестирования будет применен ко всем описанным ранее компонентам и сервису MovieService:

Блочные тесты проверяют, чтобы компоненты и сервисы требуемым образом запускались в контролируемом тестовом окружении.

В реализации тестов используются Angular Testing Utilities, которые облегчают создание тестов, а также фреймворк Karma.

Блочное тестирование будет протестировано в браузере Chrome версии 78.0.3904.108.

2.6 Стратегия интеграционного тестирования

Второй этап - интеграционное тестирование.

Блочные тесты проверяют, чтобы компоненты и сервисы правильно запускались в контролируемом тестовом окружении. Однако нет гарантии, что компоненты и сервисы будут взаимодействовать друг с другом в окружении Angular, поэтому проводится комплексное тестирование. Комплексный тест

моделирует тестирование человеком. То есть тест спроектирован взаимодействовать с нашим приложением точно так же, как это делаем мы – через интерфейс браузера.

Интеграционное тестирование будет проведено для следующих взаимодействий между компонентами и сервисами.

Схема интеграции:

- Компонент MoviesComponent и сервис MovieService.

Начало теста -> 1.1 Вызов метода ngOnInit() -> 1.2 Вызов метода getMovies() -> 1.3 Вызов метода getMovies (): Observable<Movie[]> -> Конец теста

Начало теста -> 2.1 Вызов метода add(name: string): void -> 2.2 Вызов метода addMovie (newMovie: Movie): Observable<Movie> -> Observable<Movie> -> Конец теста

Начало теста -> 3.1 Вызов метода delete(movie: Movie): void-> 3.2 Вызов метода deleteMovie (movie: Movie | number): Observable<Movie>-> Конец теста

- Компонент MovieDetailComponent и сервис MovieService.

Начало теста -> 4.1 Вызов метода ngOnInit(). -> 4.2 Вызов метода getMovie(id: number): Observable<Movie>.-> Конец теста

Начало теста -> 5.1 Вызов метода save()-> 5.2 Вызов метода updateMovie (movie: Movie): Observable<any>. -> Конец теста

- Модули AppModule, AppRoutingModule и компонентами DashboardComponent.

Начало теста ->
6.1 Импорт DashboardComponent в AppRoutingModule.

6.2 Импорт DashboardComponent в AppModule.

6.3 Импорт AppRoutingModule в AppModule. -> Конец теста

Используемые инструменты для интеграционное тестирование: Protractor, утилиты тестирования Angular. Метод проведения: автоматическое тестирование.

Интеграционное тестирование будет протестировано в веб-браузере Chrome версии 78.0.3904.108.

В данной работе выбран принцип восходящего тестирования. Сначала будет тестироваться самый нижний уровень системы. Затем постепенно к более низким уровням будут интегрироваться более высокоуровневые модули.

2.7 Стратегия аттестационного тестирования

В ходе аттестационное тестирования будет протестирована работоспособность приложения и его возможность осуществлять заявленный функционал. Будут проверяться все функциональные требования.

Аттестационное тестирование будет проводиться методом «живого человека». В роли такого человека выступает сам автор тестирования.

Тестирующий человек, по заранее заданным инструкциям (TestCases), производит требуемые действия и сверяется с заранее заданными результатами.

Для проведения аттестационного тестирования необходимо использовать веб-браузер Chrome версии 78.0.3904.108

2.8 Стратегия нагрузочного тестирования

Нагрузочное тестирование – это процесс умышленной нагрузки системы, с целью определения показателей производительности, времени отклика, проверки соответствия требованиям, которые были предъявлены к данной системе или отдельному устройству. Целью данного тестирования является оценка производительности и работоспособности тестируемого модуля. Производительность, время отклика и соответствие требованиям исследуется при отображении результата на разных нагрузках в довольно широких диапазонах, а точнее по средствам отображения списка фильмов большого объема, когда:

Критерии нагрузочного тестирования	Количество элементов в списке
Малое количество элементов в списке	1
Среднее количество элементов в списке	10
Большое количество элементов в списке	100

2.9 Способы оценивания результатов тестов

2.9.1 Критерии прохождения тестов

Любой тест считается успешно пройденным, если ожидаемый результат и фактический результат совпадают. Если тест завершается неудачей, то перед принятием решения целесообразно проверить правильность самого теста. Если тест завершился неудачей и тест реализован правильно, то представляется заключение о найденной ошибке.

2.9.2 Критерий приостановления тестирования

Тестирование должно быть приостановлено, если количество не пройденных тестов превысит 40% от общего количества.

2.9.3 Критерий возобновления работы

Необходимо заново начать тестирование при получении уведомления, что найденные при тестировании ошибки исправлены

.

3 Детальный план тестов

3.1 Корректность возвращаемых данных

Под корректной работой сервера подразумевается его доступность, а также отсутствие ошибок в логике работы сервера. Кроме того, сервер должен возвращать валидный JSON (см. ниже), коды ответа сервера должны быть в диапазоне 200-399. Пример валидного JSON-файла со списком фильмов:

[{ id: 1, name: 'Кино 1' },
{ id: 2, name: 'Кино 2'},
{ id: 3, name: 'Кино 3'}]

Валидность JSON-файла можно проверить при помощи сервиса-валидатора (например, jsonlint.com).

3.2 План блочного тестирования

DashboardComponent	
Тест	**Б1**
Цель теста (описание)	Конструктор вызывается при создании объекта класса и определяет зависимости компонента. Цель теста: определить подключена ли зависимость от MovieService.
Тип теста	Общий
Объект тестирования (модуль, интерфейс или функциональность)	Компонент: DashboardComponent Конструктор: constructor(private movieService: MovieService) { }

Входные параметры	movieService – объект класса MovieService. Используется для связывания компонента с сервисом MovieService.
Косвенные данные	MockMovieService – заглушка для MockService. Результаты работы заглушки: • метод getMovies() возвращает пустой массив типа Observable<Movie[]> • getMovie() возвращает пустой JSON объект. Подключение к Интернет отсутствует.
Ожидаемый результат	Компонент создается. Ошибка сборки компонента отсутствует: TS2304: Cannot find name MovieService.
MoviesComponent	
Тест	**Б2**
Цель теста (описание)	Конструктор вызывается при создании объекта класса и определяет зависимости компонента. Цель теста: определить, подключена ли зависимость MovieService
Тип теста	Общий
Объект тестирования (модуль, интерфейс или функциональность)	Компонент: MoviesComponent Конструктор: constructor(private movieService: MovieService) { }
Входные параметры	movieService – объект класса MovieService. Используется для связывания компонента с сервисом MovieService.
Косвенные данные	MockMovieService – заглушка для MockService. Результаты работы заглушки: • метод getMovies() возвращает пустой массив типа Observable<Movie[]> • getMovie() возвращает пустой JSON объект. Подключение к Интернет отсутствует.
Ожидаемый результат	Компонент создается. . Ошибка сборки компонента отсутствует: TS2304: Cannot find name MovieService.

MovieDetailComponent	
Тест	**Б3**
Цель теста (описание)	Конструктор вызывается при создании объекта класса и определяет зависимости компонента. Цель теста: определить, подключена ли зависимость MovieService, библиотеки ActivatedRoute и Location.
Тип теста	Общий
Объект тестирования (модуль, интерфейс или функциональность)	**Компонент**: **MovieDetailComponent** **Конструктор**: constructor (private route: ActivatedRoute, private movieService: MovieService, private location: Location)
Входные параметры	**movieService** – объект класса MovieService. Используется для связывания компонента с сервисом MovieService.
Косвенные данные	**MockMovieService** – заглушка для MockService. Результаты работы заглушки: • метод getMovies() возвращает пустой массив типа Observable<Movie[]> • getMovie() возвращает пустой JSON объект. Подключение к Интернет отсутствует..
Ожидаемый результат	Компонент создается. Ошибки сборки компонента отсутствуют: 1. TS2304: Cannot find name MovieService. 2. TS2304: Cannot find name ActivatedRoute. 3. TS2304: Cannot find name Location.
MoviesComponent	

Тест	**Б4**
Цель теста (описание)	Конструктор вызывается при создании объекта класса и определяет зависимости компонента. Цель теста: определить, подключена ли зависимость MovieService.
Тип теста	Общий.
Объект тестирования (модуль, интерфейс или функциональность)	**Компонент**: **MoviesComponent**. **Конструктор**: constructor(private route: ActivatedRoute, private movieService: MovieService, private location: Location)
Входные параметры	**movieService** – объект класса MovieService. Используется для связывания компонента с сервисом MovieService.
Косвенные данные	Подключение к сервису настроено следующим образом (используется не соответствующий сервису путь к нему): import { MovieService } from '@services/in-memory-data.service'; **MockMovieService** – заглушка для MockService. Результаты работы заглушки: • метод getMovies() возвращает пустой массив типа Observable<Movie[]> • getMovie() возвращает пустой JSON объект. У компонента есть зависимости ActivatedRoute и Location, которые работают с роутингом. В коде

	теста в качестве заглушек этих классов заглушки классов должен импортироваться RouterTestingModule. RoutingTestingModule легко решает зависимости ActivateRoute и Location в коде теста. RoutingTestingModule устанавливает маршрутизатор, который будет использоваться для тестирования, также обрабатывает другие ситуации с роутингом. Подробнее о заглушке: https://angular.io/api/router/testing/RouterTestingModule?source=post_page---------- Подключение к Интернет присутствует.
Ожидаемый результат	Компонент создается. Следующие ошибки сборки компонента отсутствуют: • TS2304: Cannot find name MovieService. • TS2304: Cannot find name ActivatedRoute. • TS2304: Cannot find name Location.
MovieDetailComponent	
Тест	**Б5**
Цель теста (описание)	При нажатии на кнопку «Назад» должен выполняться переход на предыдущую страницу, на которой был пользователь. Для выполнения теста нужно выполнить следующие шаги: 1. Зайти на страницу: http://localhost:4200/dashboard, где localhost – хост, 4200 – порт. 2. Затем перейти на страницу, где 11 – идентификатор любого существующего

	фильма в системе (предварительно создать, если отсутствует). http://localhost:4200/detail/11 3. Нажать на кнопку «Назад».
Тип теста	Общий.
Объект тестирования (модуль, интерфейс или функциональность)	**Компонент**: **MovieDetailComponent**. **Метод**: goBack().
Входные параметры	-
Косвенные данные	Подключение к Интернет отсутствует.
Ожидаемый результат	Будет выполнен переход на страницу http://localhost:4200/dashboard, где localhost – хост, 4200 – порт.
MovieService	
Тест	**Б6**
Цель теста (описание)	Тест проверяет, что метод возвращает ошибку в случае отсутствия подключения к Интернет, ошибки сервера, неверного формата JSON-файла (см. раздел 3.1 корректность возвращаемых данных),, полученного с сервера.
Тип теста	Негативный
Объект тестирования (модуль, интерфейс или функциональность)	**Сервис**: MovieService. **Метод**: getMovies (): Observable<Movie[]>
Входные параметры	Сервер возвращает ошибку 4xx-5xx.
Косвенные данные	const mockData = [{ id: 1, name: 'Кино 1' }, { id: 2, name: 'Кино 2'}, { id: 3, name: 'Кино 3'}

	] as Movie[];
	Заглушка, имитирующая url для получения сведений о фильме:
	let service;
	beforeEach(inject([MovieService], s => {
	service = s;
	}));
	const apiUrl = (id: number) => {
	return `${service.moviesUrl}/${this.mockId}`;
	};
	Подключение к Интернет отсутствует.

Ожидаемый результат	Будет выведено сообщение «Error: xxx» , где xxx – код ошибки.
Тест	**Б7**
Цель теста (описание)	Проверка получения списка фильмов.
Тип теста	Общий.
Объект тестирования (модуль, интерфейс или функциональность)	**Сервис**: MovieService. **Метод**: getMovies (): Observable<Movie[]>
Входные параметры	-
Косвенные данные	Заглушка со списком фильмов, который нужно будет получить: const mockData = [{ id: 1, name: 'Кино 1' }, { id: 2, name: 'Кино 2'}, { id: 3, name: 'Кино 3'}] as Movie[]; Заглушка, имитирующая url для получения сведений о фильме: const apiUrl = (id: number) => {

	return `${service.moviesUrl}/${this.mockId}`; }; Подключение к Интернет отсутствует.
Ожидаемый результат	Полученный объект типа Observable<Movie[]> совпадает с mockData (заглушкой) по длине и содержимому.
Тест	**Б8**
Цель теста (описание)	Проверка получения информации о фильме.
Тип теста	Общий.
Объект тестирования (модуль, интерфейс или функциональность)	**Сервис**: MovieService. **Метод**: getMovie(id: number): Observable<Movie>
Входные параметры	Id = 1
Косвенные данные	Заглушка со списком фильмов, который нужно будет получить: const mockData = [{ id: 1, name: 'Кино 1' }, { id: 2, name: 'Кино 2'}, { id: 3, name: 'Кино 3'}] as Movie[]; Подключение к Интернет отсутствует.
Ожидаемый результат	Свойство name объекта типа Observable<Movie> совпадает со значением 'Кино 1' (см. заглушку со списком фильмов).
Тест	**Б9**
Цель теста (описание)	Конструктор вызывается при создании объекта класса и определяет зависимости компонента. Цель теста: определить, подключена ли зависимость от библиотеки HttpClient.
Тип теста	Общий

Объект тестирования (модуль, интерфейс или функциональность)	**Сервис**: MovieService. **Конструктор**: constructor(private http: HttpClient) { }
Входные параметры	
Косвенные данные	HttpTestingController - позволяет имитировать и сбрасывать запросы, используется для перехвата запросов. Подробнее (описание возвращаемых значений методов): https://angular.io/api/common/http/testing/HttpTestingController
Ожидаемый результат	Компонент создается. Ошибка сборки компонента отсутствует (TS2304: Cannot find name HttpClient).
Тест	**Б10**
Цель теста (описание)	Тест проверяет, что метод возвращает ошибку в случае отсутствия подключения к Интернет, ошибки сервера, неверного формата JSON-файла (см. раздел 3.1 корректность возвращаемых данных), полученного с сервера.
Тип теста	Негативный
Объект тестирования (модуль, интерфейс или функциональность)	**Сервис**: MovieService. **Метод**: **Метод**: getMovie(id: number): Observable<Movie>
Входные параметры	Сервер возвращает ошибку 4xx-5xx.
Косвенные данные	const mockData = [{ id: 1, name: 'Кино 1' }, { id: 2, name: 'Кино 2'}, { id: 3, name: 'Кино 3'}] as Movie[];

	HttpTestingController - позволяет имитировать и сбрасывать запросы, используется для перехвата запросов. Подробнее (описание возвращаемых значений методов): https://angular.io/api/common/http/testing/HttpTestingController Подключение к Интернет отсутствует.
Ожидаемый результат	Будет выведено сообщение «Error: xxx» , где xxx – код ошибки.
Тест	**Б11**
Цель теста (описание)	Проверка обновления информации о фильме.
Тип теста	Общий
Объект тестирования (модуль, интерфейс или функциональность)	**Сервис**: MovieService. **Метод**: updateMovie (movie: Movie): Observable<any>
Входные параметры	Movie = { id: 1, name: 'Кино новое' }
Косвенные данные	const mockData = [{ id: 1, name: 'Кино 1' }, { id: 2, name: 'Кино 2'}, { id: 3, name: 'Кино 3'}] as Movie[]; Подключение к Интернет отсутствует.
Ожидаемый результат	Наименование фильма с id = 1 будет изменено на «Кино новое» в mockData[0].
Тест	**Б12**
Цель теста (описание)	Тест проверяет, что метод возвращает ошибку в случае отсутствия подключения к Интернет, ошибки сервера, неверного формата JSON-файла (см.

	раздел 3.1 корректность возвращаемых данных), полученного с сервера.
Тип теста	Негативный
Объект тестирования (модуль, интерфейс или функциональность)	**Сервис**: MovieService. **Метод**: updateMovie (movie: Movie): Observable<any>w
Входные параметры	Сервер возвращает ошибку 4xx-5xx.
Косвенные данные	const mockData = [{ id: 1, name: 'Кино 1' }, { id: 2, name: 'Кино 2'}, { id: 3, name: 'Кино 3'}] as Movie[]; Заглушка, имитирующая url для получения сведений о фильме: HttpTestingController - позволяет имитировать и сбрасывать запросы, используется для перехвата запросов. Подробнее (описание возвращаемых значений методов): https://angular.io/api/common/http/testing/HttpTestingController Подключение к Интернет отсутствует.
Ожидаемый результат	Будет выведено сообщение «Error: xxx» , где xxx – код ошибки.
Тест	**Б13**
Цель теста (описание)	Тест проверяет, что метод возвращает ошибку в случае превышения максимально допустимого количества символов (100) в наименовании фильма.
Тип теста	Негативный

Объект тестирования (модуль, интерфейс или функциональность)	**Сервис**: MovieService. **Метод**: updateMovie (movie: Movie): Observable<any>
Входные параметры	Movie = { id: 1, name: '12345678912345678901234567890123456789012345 67890222222222222222222222222222222222222222 22222222222222222222222222222222222222222222 22222222222222222222222222222222222222222222 22222222222222222222222222222222222222222222 22222222222222222222222222222222222222222222 22222222222222222222222222' }
Косвенные данные	Заглушка, имитирующая url для получения сведений о фильме: HttpTestingController - позволяет имитировать и сбрасывать запросы, используется для перехвата запросов. Подробнее (описание возвращаемых значений методов): https://angular.io/api/common/http/testing/HttpTestingController Подключение к Интернет отсутствует.
Ожидаемый результат	Будет выведено сообщение «Error: превышено максимально допустимое количество символов в поле «Наименование»».
Тест	**Б14**
Цель теста (описание)	Тест проверяет, что метод возвращает ошибку в случае незаполненного поля «Наименование».
Тип теста	Негативный
Объект тестирования (модуль, интерфейс	**Сервис**: MovieService. **Метод**: updateMovie (movie: Movie): Observable<any>

или функциональность)	
Входные параметры	Movie = { id: 1, name: '"' }
Косвенные данные	Заглушка, имитирующая url для получения сведений о фильме: HttpTestingController - позволяет имитировать и сбрасывать запросы, используется для перехвата запросов. Подробнее (описание возвращаемых значений методов): https://angular.io/api/common/http/testing/HttpTestingController Подключение к Интернет отсутствует.
Ожидаемый результат	Будет выведено сообщение «Error: Заполните поле «Наименование»».
Тест	**Б15**
Цель теста (описание)	Тест проверяет, что метод возвращает ошибку в случае незаполненного поля «Идентификатор».
Тип теста	Негативный
Объект тестирования (модуль, интерфейс или функциональность)	**Сервис**: MovieService. **Метод**: updateMovie (movie: Movie): Observable<any>
Входные параметры	Movie = { id: ‘’, name: ''пвпв” }
Косвенные данные	Заглушка, имитирующая url для получения сведений о фильме: HttpTestingController - позволяет имитировать и сбрасывать запросы, используется для перехвата запросов. Подробнее (описание возвращаемых значений методов):

	https://angular.io/api/common/http/testing/HttpTestingController Подключение к Интернет отсутствует.
Ожидаемый результаты	Будет выведено сообщение «Error: Запись не существует».
Тест	**Б16**
Цель теста (описание)	Тест проверяет добавление нового фильма.
Тип теста	Общий
Объект тестирования (модуль, интерфейс или функциональность)	**Сервис**: MovieService. **Метод**: addMovie (newMovie: Movie): Observable<Movie>
Входные параметры	Movie = { id: ‘’, name: 'Кино новое' }
Косвенные данные	Заглушка со списком фильмов: const mockData = [{ id: 1, name: 'Кино 1' }, { id: 2, name: 'Кино 2'}, { id: 3, name: 'Кино 3'}] as Movie[]; Подключение к Интернет отсутствует.
Ожидаемый результат	Добавлен новый фильм. mockData[4] = id: ‘4’, name: 'Кино новое' }.
Тест	**Б17**
Цель теста (описание)	Тест проверяет, что метод возвращает ошибку в случае отсутствия подключения к Интернет, ошибки сервера, неверного формата JSON-файла (см. раздел 3.1 корректность возвращаемых данных), полученного с сервера.
Тип теста	Негативный
Объект тестирования (модуль, интерфейс	**Сервис**: MovieService.

или функциональность)	**Метод**: addMovie (newMovie: Movie): Observable<Movie>
Входные параметры	Сервер возвращает ошибку 4xx-5xx.
Косвенные данные	HttpTestingController - позволяет имитировать и сбрасывать запросы, используется для перехвата запросов. Подробнее (описание возвращаемых значений методов): https://angular.io/api/common/http/testing/HttpTestingController Подключение к Интернет отсутствует.
Ожидаемый результат	Будет выведено сообщение «Error: xxx» , где xxx – код ошибки.
Тест	**Б18**
Цель теста (описание)	Тест проверяет, что метод возвращает ошибку в случае превышения максимально допустимого количества символов (100) в наименовании при добавлении фильма.
Тип теста	Негативный
Объект тестирования (модуль, интерфейс или функциональность)	**Сервис**: MovieService. **Метод**: addMovie (newMovie: Movie): Observable<Movie>
Входные параметры	Movie = { id: ‘’, name: '1234567891234567890123456789012345678901234567890222' }

Косвенные данные	Заглушка для списка фильмов: const mockData = [{ id: 1, name: 'Кино 1' }, { id: 2, name: 'Кино 2'}, { id: 3, name: 'Кино 3'}] as Movie[]; HttpTestingController - позволяет имитировать и сбрасывать запросы, используется для перехвата запросов. Подробнее (описание возвращаемых значений методов): https://angular.io/api/common/http/testing/HttpTestingController Подключение к Интернет отсутствует.
Ожидаемый результат	Будет выведено сообщение «Error: превышено максимально допустимое количество символов в поле «Наименование»». Объект mockData = [{ id: 1, name: 'Кино 1' }, { id: 2, name: 'Кино 2'}, { id: 3, name: 'Кино 3'}] не изменится.
Тест	**Б19**
Цель теста (описание)	Тест проверяет, что метод возвращает ошибку в случае незаполненного поля «Наименование» при добавлении записи.
Тип теста	Негативный
Объект тестирования (модуль, интерфейс или функциональность)	**Сервис**: MovieService. **Метод**: addMovie (newMovie: Movie): Observable<Movie>
Входные параметры	Movie = { id: ‘’, name: ‘”” }
Косвенные данные	Заглушка для списка фильмов:

	const mockData = [{ id: 1, name: 'Кино 1' }, { id: 2, name: 'Кино 2'}, { id: 3, name: 'Кино 3'}] as Movie[]; HttpTestingController - позволяет имитировать и сбрасывать запросы, используется для перехвата запросов. Подробнее (описание возвращаемых значений методов): https://angular.io/api/common/http/testing/HttpTestingController Подключение к Интернет отсутствует.
Ожидаемый результат	Будет выведено сообщение «Error: Заполните поле «Наименование»». Объект mockData = [{ id: 1, name: 'Кино 1' }, { id: 2, name: 'Кино 2'}, { id: 3, name: 'Кино 3'}] не изменится.
Тест	**Б20**
Цель теста (описание)	Тест проверяет удаление фильма.
Тип теста	Общий
Объект тестирования (модуль, интерфейс или функциональность)	**Сервис**: MovieService. **Метод**: deleteMovie (movie: Movie | number): Observable<Movie>
Входные параметры	Movie = { id: 1, name: 'Кино 1' }
Косвенные данные	Заглушка со списком фильмов: const mockData = [

	{ id: 1, name: 'Кино 1' }, { id: 2, name: 'Кино 2'}, { id: 3, name: 'Кино 3'}] as Movie[]; Подключение к Интернет отсутствует.
Ожидаемый результат	Фильм с идентификатором 1 будет удален. mockData = [{ id: 2, name: 'Кино 2'}, { id: 3, name: 'Кино 3'}].
Тест	**Б21**
Цель теста (описание)	Тест проверяет, что метод возвращает ошибку в случае отсутствия подключения к Интернет, ошибки сервера, некорректного формата JSON-файла (см. раздел 3.1 корректность возвращаемых данных), полученного с сервера.
Тип теста	Негативный
Объект тестирования (модуль, интерфейс или функциональность)	**Сервис**: MovieService. **Метод**: deleteMovie (movie: Movie | number): Observable<Movie>
Входные параметры	Сервер возвращает ошибку 4xx-5xx.
Косвенные данные	HttpTestingController - позволяет имитировать и сбрасывать запросы, используется для перехвата запросов. Подробнее (описание возвращаемых значений методов): https://angular.io/api/common/http/testing/HttpTestingController Подключение к Интернет отсутствует.
Ожидаемый результат	Будет выведено сообщение

	«Error: xxx» , где xxx – код ошибки.
Тест	**Б22**
Цель теста (описание)	Тест проверяет, что метод возвращает ошибку в случае незаполненного поля «Идентификатор» при удалении записи.
Тип теста	Негативный
Объект тестирования (модуль, интерфейс или функциональность)	**Сервис**: MovieService. **Метод**: deleteMovie (movie: Movie | number): Observable<Movie>
Входные параметры	Movie = { id: ‘’, name: ''ваыва” }
Косвенные данные	Заглушка для списка фильмов: const mockData = [{ id: 1, name: 'Кино 1' }, { id: 2, name: 'Кино 2'}, { id: 3, name: 'Кино 3'}] as Movie[]; HttpTestingController - позволяет имитировать и сбрасывать запросы, используется для перехвата запросов. Подробнее (описание возвращаемых значений методов): https://angular.io/api/common/http/testing/HttpTestingController Подключение к Интернет отсутствует.
Ожидаемый результат	Будет выведено сообщение “Error: Невозможно удалить запись. Идентификатор не заполнен”. Объект mockData = [{ id: 1, name: 'Кино 1' }, { id: 2, name: 'Кино 2'}, { id: 3, name: 'Кино 3'}

	] не изменится.

3.3 План интеграционного тестирования

Тест	**И1**
Цель теста (описание)	Получение списка фильма при обращении к странице «Сведения о фильме».
Условия выполнения теста	Подключение к интернету присутствует на всех этапах. Должен быть создан список из 3 фильмов: { id: 1, name: 'Кино 1' }, { id: 2, name: 'Кино 2'}, { id: 3, name: 'Кино 3'}] Начало теста -> 1.1 Вызов метода ngOnInit() -> 1.2 Вызов метода getMovies() -> 1.3 Вызов метода getMovies (): Observable<Movie[]> -> Конец теста
Тип теста	Общий
Объект тестирования (модуль, интерфейс или функциональность)	Взаимодействие между компонентом MoviesComponent и сервисом MovieService.
Входные параметры	-
Косвенные данные	Нет
Ожидаемый результат	Этап 1.1 Происходит вызов ngOnInit() Этап 1.2 Вызов метода getMovies() Этап 1.3 Компонент отображает список из 3 фильмов. Возвращаемое значение: = [{ id: 1, name: 'Кино 1' }, { id: 2, name: 'Кино 2'},

	{ id: 3, name: 'Кино 3'}]
Тест	**И2**
Цель теста (описание)	Проверка добавления фильма.
Условия выполнения теста	Подключение к интернету присутствует на всех этапах. Список фильмов должен быть пуст. Начало теста -> 2.1 Вызов метода add(name: string): void -> 2.2 Вызов метода addMovie (newMovie: Movie): Observable<Movie> -> Observable<Movie> -> Конец теста
Тип теста	Общий
Объект тестирования (модуль, интерфейс или функциональность)	Взаимодействие между компонентом MoviesComponent и сервисом MovieService.
Входные параметры	Этап 2.1. и 2.2 : {id:'',name:"Кино"}- добавляемый фильм
Косвенные данные	Нет
Ожидаемый результат	Этап 2.1 Входные параметры переданы. Этап 2.2 Возвращаемое значение: {id:'1',name:"Кино"}
Тест	**И3**
Цель теста (описание)	Удаление фильма.
Условия выполнения теста	Подключение к интернету присутствует на всех этапах. Должен быть создан фильм:

	{id:’1’,name:”Кино”}
Тип теста	Общий
Объект тестирования (модуль, интерфейс или функциональность)	Взаимодействие между компонентом MoviesComponent и сервисом MovieService. Начало теста -> 3.1 Вызов метода delete(movie: Movie): void-> 3.2 Вызов метода deleteMovie (movie: Movie | number): Observable<Movie>-> Конец теста
Входные параметры	3.1 {id:’1’,name:”Кино”} 3.2 {id:’1’,name:”Кино”}
Косвенные данные	Нет
Ожидаемый результат	Этап 3.3 Входные параметры переданы. Этап 3.2 Список становится пустым: возвращаемое значение: [].
Тест	**И4**
Цель теста (описание)	Обновление информации о фильме.
Условия выполнения теста	Начало теста -> 5.1 Вызов метода save()-> 5.2 Вызов метода updateMovie (movie: Movie): Observable<any>. -> Конец теста Должен быть создан один фильм: {id:’1’,name:”Кино”} Подключение к интернету присутствует на всех этапах.
Тип теста	Общий
Объект тестирования (модуль, интерфейс или функциональность)	Взаимодействие между компонентом MovieDetailComponent и сервисом MovieService.

Входные параметры	Этап 5.2 {id:’1’,name:”Кино2”}
Косвенные данные	Нет
Ожидаемый результат	5.1 Входные параметры переданы. 5.2 Возвращаемое значение: {id:’1’,name:”Кино2”}
Тест	**И5**
Цель теста (описание)	Получение информации о фильме.
Условия выполнения теста	Начало теста -> 4.1 Вызов метода ngOnInit(). -> 4.2 Вызов метода getMovie(id: number): Observable<Movie>.-> Конец теста. Подключение к интернету присутствует на всех этапах теста. В списке фильмов лолжен быть создан фильм со следующими значениями: {id:”1’,name:”rhrh”}
Тип теста	Общий
Объект тестирования (модуль, интерфейс или функциональность)	Взаимодействие между компонентом MovieDetailComponent и сервис MovieService.
Входные параметры	Этап 4.2 {id:”1’,name:””}
Косвенные данные	Нет
Ожидаемый результат	Этап 4.1 Входные параметры переданы. Этап 4.2 Метод возвращает {id:”1’,name:”rhrh”}
Тест	**И6**

Цель теста (описание)	Обновление информации о фильме.
Условия выполнения теста	Подключение к интернету присутствует на всех этапах теста. Список фильмов должен быть пуст. Начало теста -> 6.1 Вызов метода save()-> 6.2 Вызов метода addMovie (movie: Movie): Observable<any> -> Конец теста
Тип теста	Общий
Объект тестирования (модуль, интерфейс или функциональность)	Взаимодействие между компонентом MovieDetailComponent и сервисом MovieService.
Входные параметры	Этап 6.1:Выполянется post-запрос к сервису. Этап 6.2 {id:'', name:"hfhffh"}
Косвенные данные	Нет
Ожидаемый результат	Этап 6.1 Входные параметры переданы. Этап 6.2 Метод возвращает: {id:'1', name:"hfhffh"}
Тест	**И7**
Цель теста (описание)	Тест проверяет, что при старте приложения выполняется переадресация на панель управления.
Условия прохождения теста	Подключение к интернету присутствует на всех этапах.
Тип теста	Общий
Объект тестирования (модуль, интерфейс или функциональность)	Взаимодействие между модулями AppModule, AppRoutingModule и компонентом DashboardComponent
Входные параметры	-

Косвенные данные	7.1 Импорт MoviesComponent в AppRoutingModule. 7.2 Импорт DashboardComponent в AppRoutingModule. 7.3 Импорт AppRoutingModule в AppModule, 7.4 Запуск приложения На всех этапах, кроме 8.4 подключение к интернету отсутствует.
Ожидаемый результат	7.1 Ошибки сборки отсутствуют. 7.2 Ошибки сборки отсутствуют. 7.3 Ошибки сборки отсутствуют. 7.4 После запуска приложения автоматически выполнится переход на панель управления.

3.4 План аттестационного тестирования

При проведении аттестационного тестирования проверялся следующий набор функциональностей системы, доступных пользователю:

Тест	**A1**
Цель теста (описание)	Тест проверяет возможность перемещения между панелью управления и страницей с фильмами («Фильмы»).
Тип теста	Общий
Объект тестирования (модуль, интерфейс или функциональность)	Функциональность: Навигация между страницей «Панель управления» и «Фильмы».
Входные параметры	-
Косвенные данные	Нет
Ожидаемый результат	На странице панели управления есть ссылка на страницу «Фильмы», нажав на которую выполняется переход на страницу «Фильмы.

	На странице «Фильмы» есть ссылка на страницу «Панель управления», нажав на которую выполняется переход на страницу «Панель управления.
Тест	**A2**
Цель теста (описание)	Тест проверяет возможность перехода на страницу «Сведения о фильме» с помощью щелчка мыши на любой фильм на странице «Фильмы»..
Условия выполнения теста	Список фильмов должен быть не пуст.
Тип теста	Общий
Объект тестирования (модуль, интерфейс или функциональность)	Функциональность: Навигация между страницей «Фильмы» и «Сведения о фильме».
Входные параметры	-
Косвенные данные	Нет
Ожидаемый результат	Выполнится переход на страницу «Сведения о фильме», на который пользователь нажал мышкой на странице «Фильмы». При нажатии на кнопку «Фильмы» на странице «Сведения о фильме» выполняется переход на страницу «Фильмы».
Тест	**A3**
Цель теста (описание)	Тест проверяет возможность перемещения между панелью управления и страницей «Сведения о фильме» с помощью кнопки «Главная»

Условия выполнения теста	Список фильмов должен быть не пуст.
Тип теста	Общий
Объект тестирования (модуль, интерфейс или функциональность)	Функциональность: Навигация между страницей «Панель управления» и «Сведения о фильме».
Входные параметры	-
Косвенные данные	Нет
Ожидаемый результат	Выполнится переход на страницу «Панель управления» после нажатия на кнопку «Главная» на странице «Сведения о фильме».
Тест	**A4**
Цель теста (описание)	Добавление фильма на странице «Фильмы».
Тип теста	Общий
Объект тестирования (модуль, интерфейс или функциональность)	Функциональность: добавление фильма.
Входные параметры	В поле «Наименование» ввести текст «текст»
Косвенные данные	Нет
Ожидаемый результат	На странице «Фильмы» есть поле для ввода наименования фильма, в которое вводится текст «текст». После нажатия на кнопку добавить на той же странице в списке фильмов появляется добавленный фильм с именем «текст».
Тест	**A5**
Цель теста (описание)	Тест проверяет невозможность добавить фильм с пустым полем «Наименование» на странице «Фильмы». В поле «Наименование» не вводить текст и нажать на кнопку «Добавить».
Тип теста	Негативный

Объект тестирования (модуль, интерфейс или функциональность)	Функциональность: добавление фильма.
Входные параметры	-
Косвенные данные	Нет
Ожидаемый результат	На странице «Фильмы» появиться сообщение «Заполните это поле».
Тест	**А6**
Цель теста (описание)	Тест проверяет невозможность добавить фильм с количеством символом больше 100 на странице «Фильмы».
Тип теста	Краевой
Объект тестирования (модуль, интерфейс или функциональность)	Функциональность: Добавление фильма.
Входные параметры	В поле «Наименование» ввести текст длиной 101 символ нажать на кнопку «Добавить».
Косвенные данные	Нет
Ожидаемый результат	На странице «Фильмы» появиться сообщение «Превышена максимальная длина поля (100 символов»).
Тест	**А7**
Цель теста (описание)	Тест проверяет возможность удалить выбранный фильм на странице «Фильмы».. Выбрать любой фильм списка, нажать на крестик рядом с фильмом.
Условия выполнения теста	Должен быть создан 1 фильм.

	Должен быть пройден тест А4, иначе тест считается проваленным.
Тип теста	Общий
Объект тестирования (модуль, интерфейс или функциональность)	Функциональность: Удаление фильма.
Входные параметры	-
Косвенные данные	Нет
Ожидаемый результат	На странице «Фильмы» исчезнет запись о выбранном фильме. Затем перейти на страницу, где 11 – идентификатор удаленного фильма в системе http://localhost:4200/detail/11. На данной странице не будут отображаться сведения о фильме. Будет выведена ошибка getMovie id= 11 failed: undefined, где 11 - идентификатор удаленного фильма в системе.
Тест	**А8**
Цель теста (описание)	Тест проверяет возможность просмотра деталей фильма на странице «Сведения о фильме». Необходимо зайти на страницу: http://localhost:4200/details/x, где localhost – хост, 4200 – порт, x – идентификатор любого существующего фильма в системе.
Условия выполнения теста	Должен быть создан как минимум 1 фильм.

	Для теста необходимо будет выбрать из списка фильмов один идентификатор существующего фильма.
Тип теста	Общий
Объект тестирования (модуль, интерфейс или функциональность)	Функциональность: Просмотр списка фильмов.
Входные параметры	-
Косвенные данные	Нет
Ожидаемый результат	На странице отобразиться наименование фильма и его идентификатор.
Тест	**A9**
Цель теста (описание)	Тест проверяет возможность просмотра списка фильмов на странице «Фильмы». Зайти на страницу: http://localhost:4200/movies, где localhost – хост, 4200 – порт
Условия выполнения теста	Должен быть созданы 10 фильмов со следующими наименованиями: • 1 • 2 • 3 • 4 • 5 • 6 • 7 • 8 • 9

	• 10
Тип теста	Общий
Объект тестирования (модуль, интерфейс или функциональность)	Функциональность: Просмотр информации о фильмах.
Входные параметры	-
Косвенные данные	Нет
Ожидаемый результат	На странице отобразиться следующий список фильмов. • 1 • 2 • 3 • 4 • 5 • 6 • 7 • 8 • 9 • 10
Тест	**A10**
Цель теста (описание)	Тест проверяет возможность редактирования информации о фильме на странице «Сведения о фильме». Необходимо перейти на страницу «Фильмы». Перейти на страницу «Сведения о фильме» для фильма с наименованием «Фильм», кликнув на него мышкой.

	Изменить текст в поле наименование на «Текст». Нажать на кнопку «Сохранить».
Условия выполнения теста	Должен быть создан один фильм с наименованием «Фильм». Пройден тест А4.
Тип теста	Общий.
Объект тестирования (модуль, интерфейс или функциональность)	Функциональность: редактирование информации о фильме.
Входные параметры	-
Косвенные данные	Нет
Ожидаемый результат	Будет выполнен переход на страницу «Фильмы» В списке фильмов отобразиться фильм с наименованием «Текст» (информация о фильме будет обновлена).
Тест	**А11**
Цель теста (описание)	При нажатии на кнопку «Назад» должен выполняться переход на предыдущую страницу, на которой был пользователь. Для выполнения теста нужно выполнить следующие шаги: 4. Зайти на страницу: http://localhost:4200/dashboard, где localhost – хост, 4200 – порт. 5. Затем перейти на страницу, где 11 – идентификатор любого существующего

	фильма в системе (предварительно создать, если отсутствует). http://localhost:4200/detail/11 6. Нажать на кнопку «Назад».
Тип теста	Общий.
Объект тестирования (модуль, интерфейс или функциональность)	Функциональность: перемещение на предыдущую страницу, на которой находился пользователь со страницы «Сведения о фильме».
Входные параметры	-
Косвенные данные	Подключение к Интернет отсутствует.
Ожидаемый результат	Будет выполнен переход на страницу http://localhost:4200/dashboard, где localhost – хост, 4200 – порт.

3.5 План нагрузочного тестирования.

Тест	H1
Тип теста	Общий
Объект тестирования (модуль, интерфейс или функциональность)	Отображение списка с малым количеством записей.
Входные параметры	Список имеющий 5 записей
Косвенные данные	Нет
Ожидаемый результат	Список имеющий 5 записей отображается.
Тест	H2
Тип теста	Общий
Объект тестирования (модуль, интерфейс или функциональность)	Отображение списка со средним количеством записей.
Входные параметры	Список имеющий 20 записей

Косвенные данные	Нет
Ожидаемый результат	Список имеющий 20 записей отображается.
Тест	Н3
Тип теста	Общий
Объект тестирования (модуль, интерфейс или функциональность)	Отображение списка с большим количеством записей.
Входные параметры	Список имеющий 100 записей
Косвенные данные	Нет
Ожидаемый результат	Список имеющий 100 записей отображается.

5 Отчет о проведении тестирования

5.1 Блочное тестирование

№Теста	**Дата**	**Результат**	**Отчет об ошибке**
Б1	11.12.2019	Пройден	
Б2	11.12.2019	Пройден	
Б3	11.12.2019	Пройден	
Б4	11.12.2019	Пройден	
Б5	11.12.2019	Пройден	
Б6	11.12.2019	Пройден	
Б7	11.12.2019	Пройден	
Б8	11.12.2019	Пройден	
Б9	11.12.2019	Пройден	
Б10	11.12.2019	Пройден	
Б11	11.12.2019	Пройден	
Б12	11.12.2019	Пройден	

Б13	11.12.2019	Пройден	
Б14	11.12.2019	Пройден	
Б15	11.12.2019	Пройден	
Б16	11.12.2019	Пройден	
Б17	11.12.2019	Пройден	
Б18	11.12.2019	Провален	Отчет об ошибке 1
Б19	11.12.2019	Пройден	
Б20	11.12.2019	Пройден	
Б21	11.12.2019	Пройден	
Б22	11.12.2019	Пройден	

5.2 Интеграционное тестирование

№Теста	**Дата**	**Результат**	**Отчет об ошибке**
И1	11.12.2019	Пройден	
И1	11.12.2019	Пройден	
И1	11.12.2019	Пройден	
И1	11.12.2019	Пройден	
И1	11.12.2019	Пройден	
И1	11.12.2019	Пройден	
И1	11.12.2019	Пройден	
И1	11.12.2019	Пройден	
И1	11.12.2019	Пройден	
И1	11.12.2019	Пройден	
И1	11.12.2019	Пройден	
И1	11.12.2019	Пройден	
И1	11.12.2019	Пройден	
И1	11.12.2019	Пройден	
И1	11.12.2019	Пройден	
И1	11.12.2019	Пройден	
И1	11.12.2019	Пройден	

И1	11.12.2019	Пройден	
И1	11.12.2019	Пройден	
И1	11.12.2019	Пройден	
И1	11.12.2019	Пройден	
И1	11.12.2019	Пройден	
И1	11.12.2019	Пройден	
И1	11.12.2019	Пройден	

5.3 Аттестационное тестирование

№Теста	**Дата**	**Результат**	**Отчет об ошибке**
А1	11.12.2019	Пройден	
А2	11.12.2019	Пройден	
А3	11.12.2019	Пройден	
А4	11.12.2019	Пройден	
А5	11.12.2019	Пройден	
А6	11.12.2019	Не пройден	Отчет об ошибке 2
А7	11.12.2019	Пройден	
А8	11.12.2019	Пройден	
А9	11.12.2019	Пройден	
А10	11.12.2019	Пройден	
А11	11.12.2019	Пройден	

5.4 Нагрузочное тестирование

№Теста	**Дата**	**Результат**	**Отчет об ошибке**
Н1	11.12.2019	Пройден	
Н2	11.12.2019	Пройден	
Н3	11.12.2019	Пройден	

6 Журнал найденных ошибок

Отчет об ошибке 1

Краткое описание: Тест проверяет, что метод возвращает ошибку в случае превышения максимально допустимого количества символов (100) в наименовании при добавлении фильма.

Ожидаемый результат: Будет выведено сообщение «Error: превышено максимально допустимое количество символов в поле «Наименование»».

Объект mockData = [{ id: 1, name: 'Кино 1' },
  { id: 2, name: 'Кино 2'},
  { id: 3, name: 'Кино 3'}
] не изменится.

Фактический результат:. Сообщение об ошибке не вывелось. Объект mockData = [{ id: 1, name: 'Кино 1' },
  { id: 2, name: 'Кино 2'},
  { id: 3, name: 'Кино 3'}
  { id: 4, name: 'Кино Новое'}]

Отчет об ошибке 2

Краткое описание: Тест проверяет невозможность добавить фильм с количеством символом больше 100 на странице «Фильмы».

Ожидаемый результат: На странице «Фильмы» появиться сообщение «Превышена максимальная длина поля (100 символов»).

Фактический результат: Произошло добавление фильма с длиной больше 100 символов. Сообщение не появилось.

7 Примеры тестов и заглушки

Пример реализации блочного теста №Б7 и заглушки для списка фильмов

```
import { HttpClientTestingModule, HttpTestingController } from '@angular/commo
n/http/testing';

import { MovieService } from './movie.service';
import { MessageService } from './message.service';
import { Movie } from '@models/movie.model';

// Заглушка для списка фильмов
const mockData = [
  { id: 1, name: 'Кино 1' },
  { id: 2, name: 'Кино 2'},
  { id: 3, name: 'Кино 3'}
] as Movie[];

describe('MovieService', () => {

  let service;

// позволяет имитировать и сбрасывать запросы используя для перехвата запросов
  let httpTestingController: HttpTestingController;

  beforeEach(() => {
    TestBed.configureTestingModule({
      imports: [
        HttpClientTestingModule
      ],
      providers: [MovieService, MessageService]
    });
    httpTestingController = TestBed.get(HttpTestingController);
  });

  beforeEach(inject([MovieService], s => {
    service = s;
  }));

  beforeEach(() => {
    this.mockMovies = [...mockData]; // заглушки
    this.mockMovie = this.mockMovies[0];
    this.mockId = this.mockMovie.id;
  });

  const apiUrl = (id: number) => {
    return `${service.moviesUrl}/${this.mockId}`;
  };

  afterEach(() => {
    // после каждого теста проверяем наличие ожидающих запросов
```

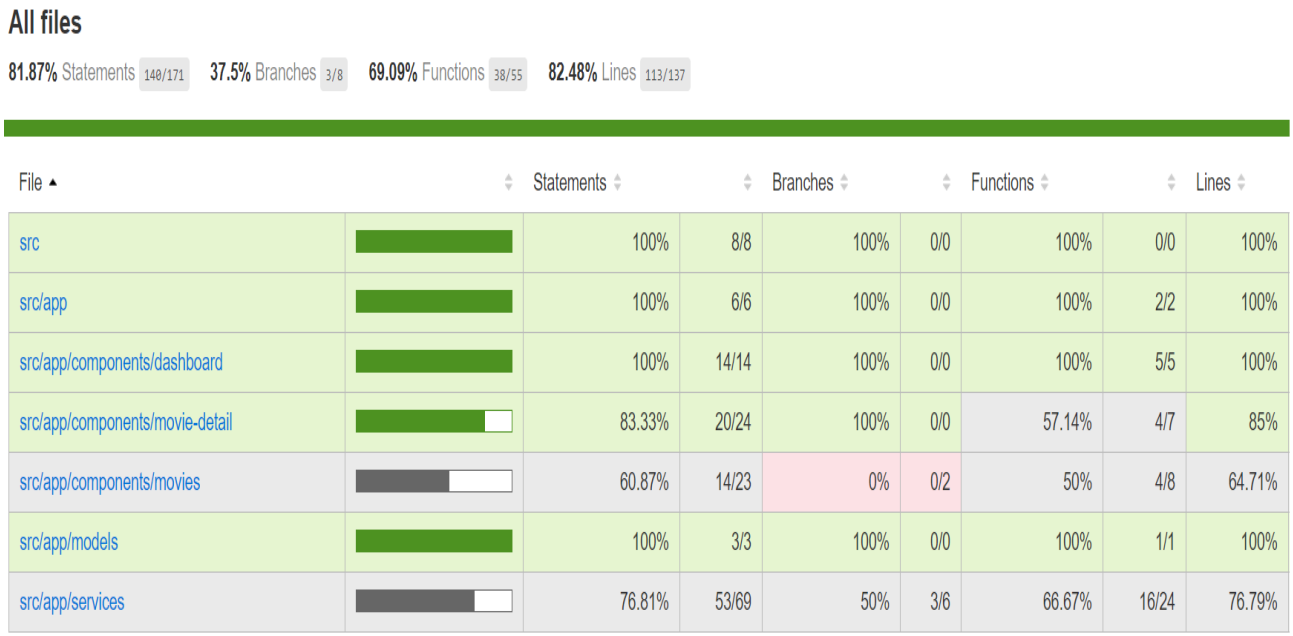
```
    httpTestingController.verify();
  });



  // при выполнеии getMovies() сервер может вернуть список фильмов, вернуть пу
стой список, выбросить ошибку, не ответить
  describe('MovieService', () => {

    it('Б7', () => {
      service.getMovies().subscribe(
        // проверяем совпадение длины и содержимого
        movies => expect(movies.length).toEqual(this.mockMovies.length),
        fail
      );
      // Отправляем GET запрос
      const req = httpTestingController.expectOne(service.moviesUrl);
      expect(req.request.method).toEqual('GET');
      // Ответ должен быть  Movie[];
      req.flush(this.mockMovies);
    });
});
```

Пример реализации интеграционного теста № И7.

```
import { AppPage } from './app.po';
import { browser  } from 'protractor';

describe('angular-movies App', () => {
  let page: AppPage; // старт-страница

  beforeEach(() => {
    page = new AppPage();
  });

// Интеграционное тестирование. Компонент dashboard и appcomponent
  it('И', async () => {
    page.navigateTo();
    const url = await browser.getCurrentUrl();
    expect(url).toContain('/dashboard'); // ожидаем эту страницу
  });
```

8 Покрытие кода тестами

Для генерации отчета по покрытию кода тестами (рис.3) использовалась команда: ng test --watch=false --code-coverage.



File	Statements		Branches		Functions		Lines	
src	100%	8/8	100%	0/0	100%	0/0	100%	
src/app	100%	6/6	100%	0/0	100%	2/2	100%	
src/app/components/dashboard	100%	14/14	100%	0/0	100%	5/5	100%	
src/app/components/movie-detail	83.33%	20/24	100%	0/0	57.14%	4/7	85%	
src/app/components/movies	60.87%	14/23	0%	0/2	50%	4/8	64.71%	
src/app/models	100%	3/3	100%	0/0	100%	1/1	100%	
src/app/services	76.81%	53/69	50%	3/6	66.67%	16/24	76.79%	

Рис.3: Покрытие кода тестами

Ниже приведены четыре различных типа отчетов о покрытии:

1. Statements- каждый оператор в программе был выполнен?
2. Branches - каждая ветвь каждой управляющей структуры была выполнена? Например, есть все операторы case, или если / else операторы были вызваны.
3. Functions- была ли вызвана каждая функция (или подпрограмма) в программе?
4. Lines- была ли выполнена каждая исполняемая строка в исходном файле?

Все типы отчетов рассчитываются в процентах, а также для первых трех выводятся количественные показатели.

В столбце Files указаны архитектурные блоки приложения.

Расчет тестового покрытия относительно исполняемого кода приложения проводится по формуле:

Covering = tested_length / code_length * 100%

• tested_length – это количество строк кода, ветвей, функций или состояний (зависит от типа отчета) покрытых тестами.

• code_length – это общее количество строк кода, ветвей, функций или состояний (зависит от типа отчета).

10 Трассируемость требований

Обозначения.

- А – аттестационный тест.
- Б – блочный тест.
- И - интеграционный тест.
- Т – требование
- ФТ – функциональное требование.
- + – требование учтено в тесте.

Т1, Т2 исключены из таблицы так как представляю собой архитектурные ограничения.

Зеленым выделены общие тесты. Красным – негативные, краевые.

Требование \ Номер теста	Т3	Т4	Т5	ФТ1	ФТ2	ФТ3	ФТ4	ФТ5
Б1	+	+						
Б2								
Б3				+				
Б4						+		
Б5			+					
Б6						+		
Б7						+		
Б8						+		

Б9						+		
Б10				+				
Б11				+				
Б12				+				
Б13								+
Б14								+
Б15								+
Б16								+
Б17								+
Б18								+
Б19								+
Б20							+	
Б21							+	
Б22								+
И1					+			
И2		+				+		
И3	+						+	
И4								+
И5			+					
И6								
И7								
А1	+							
А2		+						
А3	+							
А4								+
А5								+
А6								+
А7							+	
А8						+		
А9					+			
А10				+				
А11			+					

Заключение

Во время тестирования приложения было выявлено несколько ошибок, представленных в отчётах об ошибках №1–3. Среди найденных ошибок все являются критическими и должны быть исправлены. По результатам тестирования необходимо исправить найденные ошибки и повторно протестировать приложение.

Покрытие теста составило относительно:

- Операторов: 81,87%.
- Ветвей: 37,5%.
- Функций: 69,09%.
- Строк кода: 82.48%.

Список источников

[1] Техническая документация Angular, Архитектура Angular.Режим доступа: https://angular.io/guide/architecture ,электронный (дата обращения: 11.12.2019)

[2] Техническая документация Angular, Файловая структура Angular. Режим доступа: https://angular.io/guide/file-structure,электронный (дата обращения: 11.12.2019)

[3] Техническая документация Angular, Архитектура Angular. Что дальше? Режим доступа: https://angular.io/guide/architecture#whats-next,электронный (дата обращения: 11.12.2019)

[4] Техническая документация Angular, Файловая структура Angular.Режим доступа: https://angular.io/guide/file-structure#application-source-files,электронный (дата обращения: 11.12.2019)

[5] Техническая документация Angular, Тестирование. Режим доступа: https://angular.io/guide/testing,электронный (дата обращения: 11.12.2019)