**5 Системное тестирование**

Системное тестирование - это тестирование программного обеспечения, выполняемое на полной, интегрированной системе, с целью проверки соответствия системы исходным требованиям. Системное тестирование относится к методам тестирования чёрного ящика, и, тем самым, не требует знаний о внутреннем устройстве системы [14]. Тестирование бывает:

* функциональное тестирование;
* тестирование пользовательского интерфейса;
* тестирование совместимости;
* тестирование безопасности;
* тестирование производительности;
* автоматическое тестирование;
* интеграционное тестирование.

## 5.1 Функциональное тестирование

Функциональное тестирование - это тестирование программного обеспечения в целях проверки реализуемости функциональных требований, то есть способности программного обеспечения в определённых условиях решать задачи, нужные пользователям [15]. Функциональные требования определяют, что именно делает программное обеспечение, какие задачи оно решает.

Функциональные требования включают в себя:

* функциональная пригодность;
* точность;
* способность к взаимодействию;
* соответствие стандартам и правилам;
* защищённость.

Тестирование и отладка программы являются наиважнейшими этапами разработки любых программных продуктов. Цель этого этапа – проверка правильности и точности реализации функций, выполнение которых возлагается на данный программный продукт. В случае выявления некоторых неточностей и ошибок необходимо проведение работ по их исправлению и доработке программного продукта до требуемого уровня.

На основе функций, которые должны быть протестированы в разработанной информационной обучающей системе, составлен чек-лист (checklist), приведенный в таблице 5.1.

Таблица 5.1 – Чек-лист

|  |  |  |  |
| --- | --- | --- | --- |
| Тестируемый модуль | | Тестируемая функция | Результат |
| 1 | | 2 | 3 |
| Запуск приложения | | Функция запуска | Выполнено успешно |
| Выход из приложения | | Функция закрытия программы |
| Ввод данных | | Определение соответствие типа вводимых данных | Выполнено успешно |
| Ввод ошибочных данных | Вывод сообщении об ошибке | | Выполнено успешно |
| Сохранение и загрузка | Сохранение данных в базу данных | | Выполнена успешно |
| Чтение данных из базы данных | |

В качестве тестирования программного продукта был выбран тест-кейс (**Test Case)**.

**Test Case** - это артефакт, описывающий совокупность шагов, конкретных условий и параметров, необходимых для проверки реализации тестируемой функции или её части.

Тестирование производилось на ОС «Windws 10.

Таблица 5.2 – Тест-кейсы

|  |  |  |  |
| --- | --- | --- | --- |
| Краткое описание | Предварительные условия | Шаги для воспроизведения | Ожидаемый результат |
| Тестирование загрузки данных | В базе данных должны быть записи | Перейти на страницу, содержащую таблицу | Отображается таблица с данными из бд |
| Тестирование кнопки «Парсера» | База данных должна хранить половину данных из файла | Перейти на страницу загрузки файла.  Выбрать файл.  Загрузить файл  Перейти на страницы нагрузки, преподавателей | Новые данные, которые хранились в файла занесены в соответствующие таблицы, данные, которые уже были повторно не заносились |
| Тестирование неправильного ввода | Переход на страницу добавления нагрузки | В поля для чисел ввести буквы  Не заполнять выпадающие списки | Пользователь увидит сообщение о некорректном вводе |
| Тестирование фильтрации | В базе данных должны быть несколько преподавателей и нагрузка для их обоих | Перейти на страницу преподавателей.  Выбрать отображение нагрузки для одного из них | На странице нагрузки показываются только записи, содержащие выбранного преподавателя |

Таблица 5.3 – Набор тестов отображения информации

|  |  |  |  |
| --- | --- | --- | --- |
| Место проведения теста | Содержание теста | Ожидаемый результат | Отметка о прохождении теста |
| Все страницы | Перейти на любую из вкладок в произвольном порядке. | Корректное отображение информации, отображение того, чего ожидает пользователь. | Да |
| Все вкладки | Увеличение масштаба отображения средствами браузера. | Появление скроллов справа и снизу, корректное отображение информации. | Да |
| Все вкладки | Обновление страницы используя средства браузера. | Состояние веб-сервиса до обновления и после не изменилось. | Да |
| Меню браузера | Нажатие функциональных кнопок браузера «Назад» и «Вперед» в произвольный момент работы веб-приложения. | Корректное поведение веб-сервиса. Отображение информации. | Да |

## 5.2 Оценка безопасности

Обеспечение безопасности информационных систем представляет собой ряд мероприятий, направленных на предотвращение несанкционированных воздействий на защищаемую информацию, а также её утечки. Поскольку приложение построено на базе фреймворка Spring, вопросы безопасности берет на себя Spring Security.

Spring Security это Java/JavaEE framework, предоставляющий механизмы построения систем аутентификации и авторизации, а также другие возможности обеспечения безопасности для корпоративных приложений, созданных с помощью Spring Framework. Проект был начат Беном Алексом (Ben Alex) в конце 2003 года под именем «Acegi Security», первый релиз вышел в 2004 году. Впоследствии проект был поглощён Spring'ом и стал его официальным дочерним проектом. Впервые публично представлен под новым именем Spring Security 2.0.0 в апреле 2008 года [16].

Ключевые объекты контекста Spring Security:

SecurityContextHolder, в нем содержится информация о текущем контексте безопасности приложения, который включает в себя подробную информацию о пользователе(Principal) работающем в настоящее время с приложением. По умолчанию SecurityContextHolder используетThreadLocal для хранения такой информации, что означает, что контекст безопасности всегда доступен для методов исполняющихся в том же самом потоке. Для того что бы изменить стратегию хранения этой информации можно воспользоваться статическим методом класса SecurityContextHolder.setStrategyName(String strategy). Более подробно SecurityContextHolder.

SecurityContext, содержит объект Authentication и в случае необходимости информацию системы безопасности, связанную с запросом от пользователя.

GrantedAuthority отражает разрешения выданные пользователю в масштабе всего приложения, такие разрешения (как правило называются «роли»), например ROLE\_ANONYMOUS, ROLE\_USER, ROLE\_ADMIN.

UserDetails предоставляет необходимую информацию для построения объекта Authentication из DAO объектов приложения или других источников данных системы безопасности. Объект UserDetailsсодержит имя пользователя, пароль, флаги: isAccountNonExpired, isAccountNonLocked, isCredentialsNonExpired, isEnabled и Collection — прав (ролей) пользователя.

UserDetailsService, используется чтобы создать UserDetails объект путем реализации единственного метода этого интерфейса.

UserDetails loadUserByUsername(String username) throws UsernameNotFoundException;

Позволяет получить из источника данных объект пользователя и сформировать из него объект UserDetails который будет использоваться контекстом Spring Security.

**5.3 Тестирование производительности**

Тестирование производительности**-** это тестирование, которое проводится для определения скорости работы системы или её части при заданной нагрузке. Тестирование производительности стремится учесть производительность уже на стадии проектирования и моделирования и системы, до начала основной стадии разработки.

Тестирование производительности служит таким типичным целям:

* для демонстрации того, что система удовлетворяет критериям производительности;
* для определения производительность какой из двух или нескольких систем лучше;
* для определения, какой элемент нагрузки или часть системы приводит к снижению производительности.
* Для оценки времени загрузки страниц системы был использован специализированный веб-сервис «https://developers.google.com». PageSpeed Insights анализирует содержание веб-страницы и предлагает решения, которые позволят ускорить ее загрузку [17].

В процессе тестирования были произведены замеры:

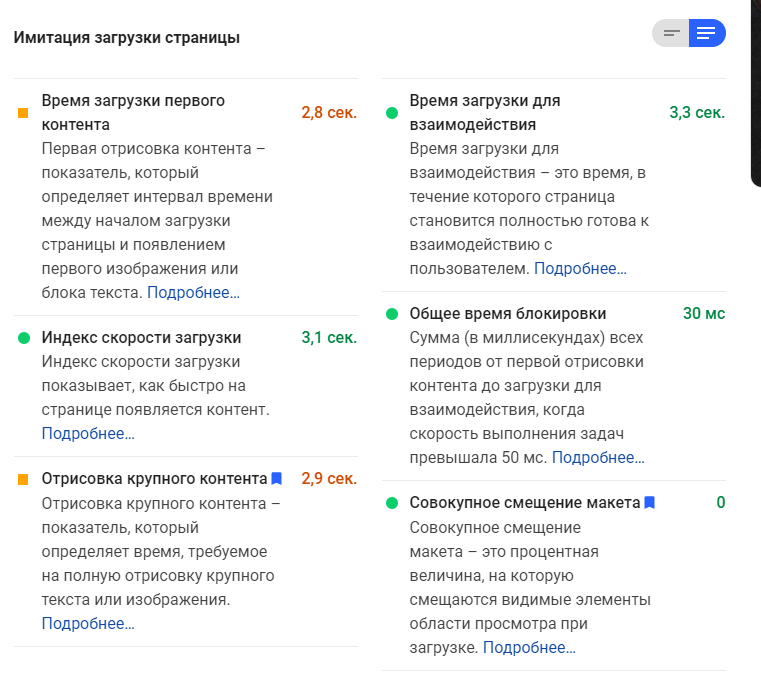
****

Рисунок 5.1 - Тестирование