

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Политехнический институт
Факультет информационных
технологий и электроники

Кафедра
«Информационная безопасность
систем и технологий»

Специальность 10.05.03 –

Специализация -

Информационная безопасность
автоматизированных систем
Защищенные автоматизированные
системы управления

ДИПЛОМНЫЙ ПРОЕКТ

на тему

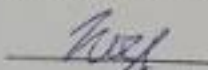
ВЕБ-САЙТ, РЕАЛИЗОВАННЫЙ С ИСПОЛЬЗОВАНИЕМ LARAVEL PHP-
ФРЕЙМВОРКА, С ЗАЩИТОЙ ОТ УГРОЗ, НАПРАВЛЕННЫХ НА
ЭКСПЛУАТАЦИЮ УЯЗВИМОСТЕЙ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Студент

Руководитель

Нормоконтролёр

Рецензент начальник сектора АО «ПНИЭИ»

 Козин А. А.
 Савенко П. Ю.
 Иванов А. П.
 Зорин А. Н.

Работа допущена к защите (протокол заседания кафедры от 28.06.21 № 16)

Заведующий кафедрой

 С.Л. Зефирова

Работа защищена с отметкой отлично (протокол заседания ГЭК
от 28.06.21 № 5)

Секретарь ГЭК

 А.П. Иванов

Пенза 2021

Оглавление

Обозначения и сокращения.....	4
Введение.....	5
1 Анализ возможных угроз веб-сайта	6
1.1 Составление списка угроз.....	6
1.2 Инъекционная атака	6
1.3 Подбор пароля.....	7
1.4 Перехват данных клиента и сервера при обмене	8
1.5 Межсайтовый скриптинг (XSS)	9
1.6 Межсайтовая подделка запроса.....	10
2 Разработка защитных мер от угроз	12
2.1 Инъекционная атака	12
2.2 Подбор пароля.....	12
2.3 Перехват данных клиента и сервера при обмене	13
2.4 Межсайтовый скриптинг (XSS)	13
2.5 Межсайтовая подделка запроса.....	14
3 Разработка программного обеспечения и конфигурации веб-сайта, которые учитывают предложенные рекомендации по защите.....	16
3.1 Анализ и выбор фреймворков и CMS.....	16
3.2 Установка Laravel и подключение git.....	19
3.3 Разработка модуля авторизации и регистрации	21
3.4 Разработка административной панели	24
3.5 Разработка страницы показа товаров.....	33
3.6 Разработка детальной страницы товара	36
3.7 Разработка заказа	37
3.8 Разработка страницы предыдущих заказов.....	43
3.9 Разработка страницы обратной связи	46
3.10 Разработка страницы показа отзывов	48

3.11	Развертывание сайта на сервере и его настройка	50
3.12	Подключение SSL для HTTPS.....	51
4	Функциональное тестирование разработанного веб-сайта от рассмотренных угроз.....	54
4.1	Инъекционная атака	54
4.2	Подбор пароля.....	54
4.3	Перехват данных клиента и сервера при обмене	55
4.4	Межсайтовый скриптинг (XSS)	56
4.5	Межсайтовая подделка запроса.....	56
5	Вредные психофизиологические факторы, влияющие на нервную систему специалиста по защите информации. Методы и средства физической культуры, снижающие их воздействие	57
	Заключение	63
	Список используемых источников.....	64

Обозначения и сокращения

XSS — Cross-Site Scripting;

БД — база данных;

JS — JavaScript;

OWASP — Open Web Application Security Project;

URL — Uniform Resource Locator;

CSRF — Cross-Site Request Forgery;

API — Application programming interface;

DOM — Document Object Model;

IP — Internet Protocol;

HTML — HyperText Markup Language;

SQL — Structured Query Language;

CMS — Content Management System;

CSS — Cascading Style Sheets;

HTTP — HyperText Transfer Protocol;

HTTPS — HyperText Transfer Protocol Secure;

TLS — Transport Layer Security;

SSL — Secure Sockets Layer;

SPA — Single Page Application.

Введение

Всемирная сеть Интернет стала неотъемлемой частью жизни почти каждого человека, почти каждая компания имеет свой сайт в интернете.

Работа сайтов может нарушаться злоумышленниками, как показывает многолетняя практика, 99% атак происходят из-за ошибок валидации входящих данных, либо уязвимостей компонентов программного обеспечения, либо по халатности системных администраторов, использующих настройки и пароли, установленные по умолчанию.

Обеспечение информационной безопасности веб-сайтов является актуальной задачей.

Целями выпускной квалификационной работы являются анализ угроз, направленных на эксплуатацию уязвимостей программного обеспечения веб-сайта, разработка методов защиты от проанализированных угроз и создание примера веб-сайта с использованием разработанных методов защиты.

Рассмотрены следующие угрозы и методы защиты от них:

- инъекционная атака;
- подбор пароля;
- перехват данных клиента и сервера при обмене;
- межсайтовый скриптинг (XSS);
- межсайтовая подделка запроса.

1 Анализ возможных угроз веб-сайта

1.1 Составление списка угроз

Классификацией векторов атак и уязвимостей занимается сообщество OWASP. Это международная некоммерческая организация, сосредоточенная на анализе и улучшении безопасности программного обеспечения.

OWASP создал список из десяти самых опасных векторов атак на Web-приложения, этот список получил название OWASP TOP-10 [1] и в нем сосредоточены самые опасные уязвимости, которые могут привести к финансовым потерям или подрыву деловой репутации, вплоть до потери бизнеса. На основе этих уязвимостей автором был составлен список наиболее актуальных угроз:

- инъекционная атака;
- подбор пароля;
- перехват данных клиента и сервера при обмене;
- межсайтовый скриптинг (XSS);
- межсайтовая подделка запроса.

1.2 Инъекционная атака

При инъекционной атаке злоумышленник внедряет недопустимые данные в веб-приложение с намерением заставить его сделать нечто, для чего приложение не было разработано/запрограммировано.

Наиболее распространённой разновидностью этой угрозы является внедрение кода через SQL-запрос, использующий ненадёжные данные. Один из примеров от OWASP вы можете увидеть ниже:

```
String query = "SELECT * FROM accounts WHERE custID = '" +  
request.getParameter("id") + "'";
```

Этот запрос можно использовать, вызвав веб-страницу с URL-адреса `http://example.com/app/accountView?id='` или `1'='1`, что приводит к возврату всех строк, хранящихся в таблице базы данных.

Главной причиной этой угрозы, связанной с внедрением кода, является отсутствие проверки и очистки данных, используемых веб-приложением. Это означает, что подобная угроза может быть практически в любом типе технологий.

Всё, что принимает параметры в качестве входных данных, потенциально может быть уязвимо для атаки путём внедрения кода.

Инъекционные атаки представляют собой серьёзный риск для владельцев сайтов, так как используют лазейки в системе безопасности для кражи конфиденциальной информации.

1.3 Подбор пароля

Это угроза, использующая уязвимость - некорректная аутентификация, может позволить злоумышленнику получить контроль над любой учётной записью в системе или, что ещё хуже, полный контроль над системой.

Сайты с нарушенной аутентификацией очень распространены в сети. Нарушение аутентификации обычно связано с логическими проблемами, возникающими в механизме аутентификации приложения, такими как плохое управление сессией, при котором происходит перечисление имён пользователей. В этом случае злоумышленник использует методы перебора, чтобы угадать данные пользователей системы.

Второй наиболее распространённой формой этой уязвимости является разрешение пользователям использовать подбор комбинации имени пользователя и пароля для этих страниц.

Согласно OWASP Top-10, подобная уязвимость может проявляться во многих формах. Веб-приложение содержит неисправную аутентификацию,

если оно:

- допускает автоматизированные атаки, такие как подбор или автоматизированное заполнение учётных данных;
- разрешает использование стандартных, слабых или хорошо известных паролей, таких как Password1 или admin/admin;
- использует слабые или неэффективные процессы восстановления учётных данных и забытых паролей, в том числе «ответы на основе знаний»;
- не использует многофакторную аутентификацию;
- предоставляет идентификаторы сеанса в URL;
- не меняет идентификаторы сеанса после успешного входа в систему;
- не делает недействительными идентификаторы сеанса. Сеансы пользователей или токены аутентификации (в частности, токены единого входа) не аннулируются должным образом при выходе из системы или в период бездействия.

Написание небезопасного программного обеспечения, содержащего подобные уязвимости, можно объяснить многими факторами, например отсутствием опыта у разработчиков, отсутствием требований к безопасности или спешка с выпуском программного обеспечения.

1.4 Перехват данных клиента и сервера при обмене

Угроза перехвата данных клиента и сервера использует то, что веб-приложения не защищают критичные данные. Незащищённость критичных данных является одной из наиболее распространённых уязвимостей. Речь идёт в первую очередь о конфиденциальных данных, которые требуют особого внимания и защиты.

Для любой организации жизненно важно понимать значимость защиты информации и конфиденциальности пользователей. Все компании должны

соблюдать местные законы о конфиденциальности.

Ответственное отношение к сбору и обработке конфиденциальных данных стало более заметным после появления Общего регламента по защите данных. Этот новый закон о конфиденциальности данных, вступивший в силу в 2018 году, определяет, как компании должны собирать, изменять, обрабатывать, хранить и удалять персональные данные.

Отсутствие шифрования конфиденциальных данных — основная причина того, что эти атаки всё ещё происходят так часто.

1.5 Межсайтовый скриптинг (XSS)

Межсайтовый скриптинг (XSS) — широко распространённая угроза, которая затрагивает многие веб-приложения. XSS-атаки состоят из внедрения вредоносных клиентских скриптов на сайт с последующим использованием сайта в качестве распространителя.

Риски, связанные с XSS, заключаются в том, что он позволяет злоумышленнику внедрять контент на сайт и изменять способ его отображения, заставляя тем самым браузер жертвы выполнять при загрузке страницы код, предоставленный злоумышленником.

Угроза XSS присутствует примерно в двух третях всех приложений.

Как правило, угрозы XSS требуют определённого типа взаимодействия со стороны пользователя, которое должно быть инициировано либо с помощью социальной инженерии, либо через посещение определённой страницы. Если уязвимость XSS не исправлена, она может быть очень опасной для любого сайта.

Согласно OWASP Top 10, существует три типа межсайтового скриптинга:

— отражённый XSS: приложение или API включает в себя непроверенный и неэкранированный пользовательский ввод как часть вывода HTML. Успешная атака позволит злоумышленнику загрузить произвольный

HTML и выполнить JavaScript в браузере жертвы. Как правило, пользователю необходимо провзаимодействовать с какой-либо вредоносной ссылкой, которая указывает на страницу, контролируруемую злоумышленником.

— сохранённый XSS: приложение или API хранит неочищенный пользовательский ввод, который позже просматривается другим пользователем или администратором. Сохранённый XSS-код часто считается высоким или критическим риском.

— DOM XSS: JavaScript-фреймворки, одностраничные приложения и API, которые динамически включают данные, контролируемые злоумышленником, уязвимы для DOM XSS.

1.6 Межсайтовая подделка запроса

CSRF — это атака, которая обманом заставляет жертву отправить злонамеренный запрос. Он наследует личность и привилегии жертвы для выполнения нежелательной функции от имени жертвы. Для большинства сайтов запросы браузера автоматически включают любые учетные данные, связанные с сайтом, такие как файл cookie сеанса пользователя, IP-адрес, учетные данные домена Windows и т. Д. Следовательно, если пользователь в настоящее время аутентифицирован на сайте, у сайта не будет возможности отличить поддельный запрос, отправленный жертвой, и законный запрос, отправленный жертвой.

CSRF-атаки нацелены на функциональность, которая вызывает изменение состояния на сервере, например, изменение адреса электронной почты или пароля жертвы или покупка чего-либо. Принуждение жертвы к извлечению данных не приносит пользы злоумышленнику, потому что злоумышленник не получает ответа, а жертва получает. Таким образом, CSRF-атаки направлены на запросы с изменением состояния.

Иногда возможно сохранить CSRF-атаку на самом уязвимом сайте. Такие

уязвимости называются «хранимыми недостатками CSRF». Этого можно достичь, просто сохранив тег IMG или IFRAME в поле, которое принимает HTML, или с помощью более сложной атаки межсайтового сценария. Если атака может сохранить CSRF-атаку на сайте, серьезность атаки увеличивается. В частности, вероятность увеличивается, потому что жертва с большей вероятностью просмотрит страницу, содержащую атаку, чем какую-либо случайную страницу в Интернете. Вероятность также увеличивается, потому что жертва обязательно уже аутентифицирована на сайте.

2 Разработка защитных мер от угроз

2.1 Инъекционная атака

Для любых остаточных динамических запросов экранировать специальные символы, используя синтаксис экранирования для этого интерпретатора.

Можно использовать положительную проверку («белый список») входных данных на стороне сервера. Это, однако, не даёт полной защиты, поскольку многим приложениям требуются специальные символы.

На сайте будет использоваться метод с экранированием специальных символов. Фреймворк Laravel, при использовании его функции для обращения к базе данных, сам экранирует специальные символы. Однако можно использовать функцию «gaw», что позволяет писать SQL-запрос.

2.2 Подбор пароля

По возможности внедрять многофакторную аутентификацию, чтобы предотвратить автоматизированные атаки злоумышленников.

Не использовать учётные данные по умолчанию, особенно это касается учётных данных администраторов.

Внедрить проверку надёжности паролей, включая тестирование новых или изменённых паролей по списку из 10 000 худших паролей.

Согласовать политику длины, сложности и ротации паролей с рекомендациями NIST800-63 В или другими современными политиками безопасности паролей.

Ограничивать или задерживать по времени неудачные попытки входа в систему. Регистрировать все сбои и предупреждать администраторов при обнаружении вброса учётных данных, их перебора или других атак.

Для сайта были выбраны 2 метода: ограничение по количеству входов, для этого у фреймворка Laravel есть специальные функции, которые позволяют ограничить количество попыток, и установка правила, что минимальный пароль – 8 символов.

2.3 Перехват данных клиента и сервера при обмене

Обязательно шифровать все конфиденциальные данные, находящиеся в состоянии покоя.

Обеспечивать наличие современных и надёжных стандартных алгоритмов, протоколов и ключей; используйте надлежащее управление ключами.

Шифровать все передаваемые данные с помощью безопасных протоколов, таких как TLS, с безопасными параметрами и приоритизацией шифрования сервером.

Хранить пароли, используя мощные адаптивные и «солёные» функции хеширования, например Argon2, scrypt, bcrypt или PBKDF2.

На сайте будут использоваться 2 метода: хранение паролей в базе данных в хешированном виде и использование протокола https для передачи данных между клиентом и сервером.

2.4 Межсайтовый скриптинг (XSS)

Использовать фреймворки, которые благодаря своей «конструкции» автоматически избегают XSS. Необходимо изучить ограничения XSS-защиты каждого фреймворка и примите соответствующие меры.

Экранирование ненадёжных данных HTTP-запроса на основе контекста в выходных данных HTML (тело, атрибут, JavaScript, CSS или URL) позволит устранить отражённые и сохранённые уязвимости XSS.

Применение контекстно-зависимого кодирования при изменении документа браузера на стороне клиента действует против DOM XSS.

Так как эта угроза нацелена на браузер клиента, а не на сервер, то защита от данной угрозы должна быть в JS фреймворке, а не в Laravel. На стороне клиента будет использоваться JS фреймворк Vue и он позволяет выводить уже экранированные данные, которые приходят с сервера.

2.5 Межсайтовая подделка запроса

Эффективным и общепринятым на сегодня способом защиты от CSRF-Атаки является токен. Токены CSRF должны создаваться на стороне сервера. Их можно сгенерировать один раз за сеанс пользователя или для каждого запроса. Токены на запрос более безопасны, чем токены на сеанс, поскольку диапазон времени, в течение которого злоумышленник может использовать украденные токены, минимален. Однако это может вызвать проблемы с удобством использования. Например, предыдущая страница может содержать недействительный токен. Взаимодействие с этой предыдущей страницей приведет к ложному положительному событию безопасности CSRF на сервере. В реализации токена для каждого сеанса после первоначального создания токена значение сохраняется в сеансе и используется для каждого последующего запроса, пока сеанс не истечет.

Когда запрос выдается клиентом, сервер должен проверить наличие и действительность токена в запросе по сравнению с токеном, найденным в пользовательском сеансе. Если токен не был найден в запросе или предоставленное значение не соответствует значению в пользовательском сеансе, запрос должен быть прерван, сеанс пользователя завершен, а событие зарегистрировано как потенциальная атака CSRF.

Токены CSRF должны быть:

- уникальные для каждого пользовательского сеанса;

- секретные;
- непредсказуемые (большое случайное значение, созданное безопасным методом).

Токены CSRF предотвращают CSRF, потому что без токена злоумышленник не может создавать допустимые запросы к внутреннему серверу.

3 Разработка программного обеспечения и конфигурации веб-сайта, которые учитывают предложенные рекомендации по защите

3.1 Анализ и выбор фреймворков и CMS

CMS – софт, используемый для создания сайта и управления его содержимым. С помощью CMS осуществляется наполнение и редактирование материалов. Любая CMS подобна детскому конструктору: она имеет множество разных по функционалу деталей, komponуя которые, можно добиться желаемого результата.

Каждая система имеет свои возможности и варианты дополнений, но все они ограничены рамками готового шаблона. Бывают узкоспециализированные CMS – укомплектованные интернет-магазины или блоги. Некоторые системы универсальны: их можно адаптировать под любой проект. Они могут быть платными и на бесплатной основе, но дополнительные функции и плагины в любом случае потребуют финансовых вложений.

Наиболее популярные бесплатные CMS:

- Joomla – довольно простая в изучении и использовании. Имеет большое количество готовых модулей не всегда хорошего качества. Часто обновляется;

- WordPress – наиболее доступная система, легкая в изучении и применении. Огромное количество платных и бесплатных плагинов не всегда могут похвастаться корректной работой;

- Drupal – универсальная CMS, требующая определенных знаний и опыта работы;

- MODx – достаточно эффективная система. Подходит для умелого подхода грамотного специалиста;

- Open Cart – специализированная программа для создания интернет-магазинов. Имеет готовые сборки, но требует определенных навыков в

настройке.

Часто используемые платные CMS:

- 1С Битрикс – CMS от 1С, что обещает свободную интеграцию между этими программами. Однако разные модификации и версии не всегда работают совместно, может потребоваться их нестандартная интеграция;

- UMI.CMS – имеет несколько основных направлений, по которым созданы готовые сборки. Адаптируется под мобильные устройства.

Плюсы CMS:

- скорость – готовый работающий шаблон с определенным набором функций можно создать за полчаса;

- простота – работа с CMS доступна даже программисту с небольшим опытом работы;

- доступность – невысокая планка специалиста делает создание сайтов с помощью CMS недорогим. В то же время, при расширении функционала может понадобиться дорогостоящая помощь специалиста;

- схожесть – создав в выбранной CMS один проект, Вы с легкостью сможете настроить следующий сайт. А имея соответствующий опыт, разберетесь и в работе других систем, поскольку функционал разных CMS перекликается друг с другом;

Недостатки CMS:

- шаблонность – одинаковые блоки предоставляются всем пользователям. Разница лишь в том, как их совместить. Впрочем, многие CMS предлагают готовые сборки для создания сайтов определенного назначения и тематики, что по-своему удобно, но ничем не выделит на фоне других;

- ограниченность – готовые детали конструктора решают свой ограниченный функционал.

- неоптимизированный код – некоторые CMS генерируют ненужные строки кода, которые изрядно замусоривают конечный результат. Чистить код от лишнего – дело долгое и требует определенной подготовки, поэтому

зачастую получается совершенно неоптимизированный код. Это увеличивает время загрузки сайта и понижает его выдачу в поисковых системах.

— отсутствие гибкости – необходимо сразу создать структуру, подходящую контенту и целям. В дальнейшем можно вносить только незначительные изменения, а любой шаг в сторону уникальности превратиться в головную боль.

Фреймворк – это платформа для создания сайтов в виде набора библиотек. Если CMS соответствует конструктору, то фреймворк скорее является 3D принтером, который позволяет воплотить в жизнь любую функциональную деталь. Существуют каркасы Framework для разных языков программирования: Java, Python, Ruby и т.д.

Виды бесплатных фреймворков, которые работают на PHP:

— Yii – имеет большое количество библиотек, что позволяет экономить время разработки, поэтому используется преимущественно для больших проектов;

— Zend Framework – данный софт позволяет свободно редактировать отдельные части проекта, благодаря их минимальной взаимозависимости;

— Laravel и CodeIgniter – имеет качественный исходный код и широкую подборку готовых решений;

Плюсы и минусы фреймворков:

— высокая производительность – сайты загружаются быстро и оперативно реагируют на производимые действия;

— уникальность – фреймворк позволяет выполнить любое требование заказчика, реализовать самые смелые задумки;

— безопасность – процесс сборки сайта с помощью фреймворка дает возможность обнаружить и устранить все потенциально опасные дыры в безопасности;

— требовательность к исполнителю – для качественной работы с библиотеками нужны квалификация и опыт. Индивидуальный подход и

решение нестандартных задач может потребовать определенное время и стоить дороже шаблонных решений.

Так как для защищенного сайта наиважнейшим приоритетом является безопасность, то между CMS и фреймворком был выбран фреймворк, а из доступных фреймворков был выбран Laravel.

3.2 Установка Laravel и подключение git

Был создан репозиторий на GitLab и сканирован на локальный сервер для разработки. Затем был установлен фреймворк Laravel [2]. Его структура представлена на рисунке 1.

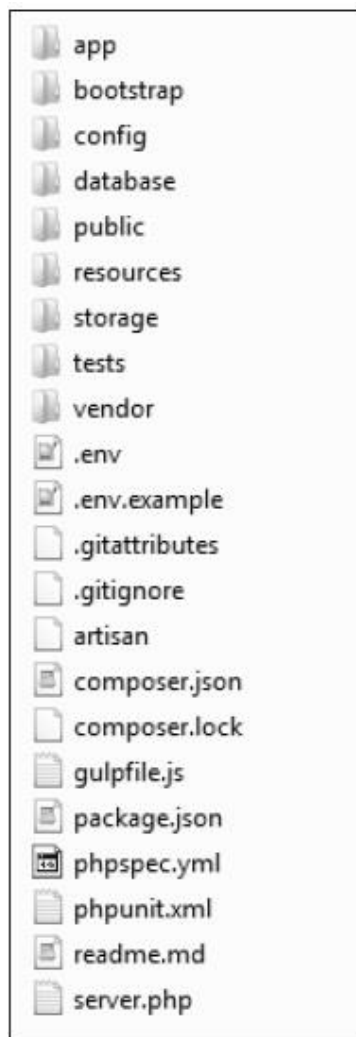


Рисунок 1 – структура Laravel

Предназначение папок в структуре:

- app – эта папка содержит основной код приложения;
- bootstrap – скрипт начальной загрузки приложения;
- config – эта папка содержит файлы конфигурации приложения;
- database — эта папка содержит перенесенную базу данных и посе́вы;
- public — это корневая папка документа приложения. Из нее запускается приложение Laravel. Она также содержит ресурсы приложения, такие как JavaScript, CSS, изображения и т. д;
- resources – эта папка содержит исходные ресурсы, такие как файлы LESS и Sass, файлы локализации и языков и Шаблоны, которые отображаются как HTML;
- storage – эта папка содержит хранилище приложения, например, загруженные файлы и т. д. Хранилище для платформы (кэш) и журналы, созданные приложением;
- test – эта папка содержит различные тестовые примеры;
- routes - эта папка содержит различные маршруты;
- vendor – эта папка содержит зависимости композитора;

Предназначение папок в основном каталоге приложения app:

- Console — все команды мастера хранятся в этой папке;
- Exceptions — та папка содержит обработчик исключений приложения, и в ней также можно хранить любые исключения, создаваемые приложением;
- Http — эта папка содержит контроллеры, фильтры и запросы;
- Policies — эта папка содержит различные политики, которые можно установить для приложения;
- Providers – эта папка содержит провайдеры служб.

Затем, с помощью инструментов npm и composer, были установлены

дополнительные зависимости. Следующим шагом необходимо настроить подключение к БД. В файле `.env` прописываются данные БД:

```
DB_CONNECTION=pgsql
DB_HOST=localhost
DB_PORT=5432
DB_DATABASE=diplom
DB_USERNAME=postgres
DB_PASSWORD=111
```

Для сборки проекта используется сборщик `webpack`, который собирает все файлы из папки `resources`, объединяется их и копирует в папку `public`.

В `Laravel` используются маршруты, которые объявлены в папке `routes`. Благодаря системе маршрутов обеспечивается защита от угрозы `Path Traversal`.

3.3 Разработка модуля авторизации и регистрации

В качестве системы аутентификации используется `Sanctum` [3]. `Sanctum` использует встроенные в `Laravel` службы аутентификации сессии на основе файлов `Cookies`. Такой подход к аутентификации обеспечивает преимущества защиты от `CSRF`, аутентификации сессии, а также защищает от утечки учетных данных аутентификации через `XSS`.

Затем создаются таблицы пользователей, ролей пользователей и привилегий для разных пользователей. Было создано 2 роли: пользователь и администратор. Привилегии указываются в конфигурационном файле `privileges.php`. И затем при необходимости могут быть проверены при необходимости в любом контроллере.

Были реализованы формы авторизации и регистрации для пользователя. Для примера представлена форма авторизации:

```
<div class="page__auth auth auth--sm">
  <h1 class="auth__title">Авторизация</h1>
  <form method="POST" action="{{ route('login') }}"
class="form auth__inner">
    {{ csrf_field() }}
    <div class="form__item form__item--center">
```

```

        <a href="{{ route('register') }}"
class="auth__link">Зарегистрироваться</a>
    </div>
    <label class="form__item">
        <span class="form__label form__label--
hidden">E-mail</span>
        <input type="email" class="input input--
middle input--default" placeholder="E-mail"
name="email" value="{{ old('email') }}" required>
        @if ($errors->has('email'))
            <span class="notice-message notice-
message--error">{{ $errors->first('email') }}</span>
        @endif
    </label>

    <label class="form__item">
        <span class="form__label form__label--
hidden">Введите пароль</span>
        <input type="password" class="input input--
middle input--default"
name="password" placeholder="Введите
пароль" required>
    </label>
    @if ($errors->has('password'))
        <span class="notice-message notice-message--
error">{{ $errors->first('password') }}</span>
    @endif
    <div class="form__item form__item--row">
        <div class="form__inner">
            <label class="checkbox">
                <input
name="remember"
class="checkbox__input
visually-hidden" {{ old('remember') ? 'checked' : '' }}>
                <span class="checkbox__item"></span>
                <span
class="checkbox__label">Запомнить меня</span>
            </label>
        </div>
    </div>

    <div class="form__item form__item--center">
        <button class="btn btn--main btn--middle
form__button">Войти</button>
    </div>

```

```
        </form>
    </div>
```

Для защиты от CSRF используются токены, в форме авторизации видно функцию, которая использует токен: `csrf_field()`.

После ввода данных посылается POST запрос в контроллер `LoginController.php`, в котором используется функция `login`. Код функции представлен ниже:

```
Validator::make($request->all(), [
    'email' => 'required|email',
    'password' => 'required'
])->validate();

if ((new AuthService)->login($request-
>input('email'), $request->input('password'))) {
    $this->clearLoginAttempts($request);
    return $this->sendLoginResponse($request);
}

if ($this->hasTooManyLoginAttempts($request)) {
    $this->fireLockoutEvent($request);
    return $this->sendLockoutResponse($request);
}

$this->incrementLoginAttempts($request);
return $this->sendFailedLoginResponse($request);
```

Сначала данные проходят валидацию, затем, если она пройдена, проверяется правильность введенных данных. Если данные не правильные, то посылается сообщение о неправильных данных. Также здесь есть функция `incrementLoginAttempts`, которая считает количество неправильных попыток входа и блокирует вход после некоторого числа попыток, за это условие отвечает функция `hasTooManyLoginAttempts`.

Форма регистрации имеет похожую структуру, но отправляет POST запрос на контроллер `RegisterController.php`, в которой используется функция `register`. Код функции представлен ниже:

```
public function register(Request $request)
{
    $this->validator($request->all())->validate();
```

```

        event(new Registered($user = $this->create($request-
>all())));
        $this->guard()->login($user);
        if ($response = $this->registered($request, $user)) {
            return $response;
        }
        return $request->wantsJson()
            ? new JsonResponse(['redirect' => $this-
>redirectTo, 'user' => $user], 201)
            : redirect($this->redirectPath()); }

```

Сначала данные проходят валидацию. Код функции validator представлен ниже:

```

return Validator::make($data, [
    'name' => ['required', 'string', 'max:255'],
    'phone' => ['string',
'max:255', 'unique:users,phone', 'nullable'],
    'email' => ['string', 'email', 'max:255',
'unique:users,email', 'nullable'],
    'password' => ['required', 'string', 'min:8',
'max:255', 'confirmed']
]);

```

Из этой функции можно узнать, что пароль должен состоять минимум из 8 символов, что усложняет подбор пароля.

Затем, если валидация данных прошла успешно, то данные о новом пользователе сохраняются в таблице БД users. Пароль сохраняется в виде хеша, чтобы, в случае взлома БД, нельзя было узнать пароли пользователей.

3.4 Разработка административной панели

Административная панель является отдельным SPA приложением внутри проекта, отделенная от пользовательской части сайта. У нее своя форма авторизации, своя маршрутизация на Vue-router, что позволяет не перезагружать страницу при переходе на другие пункты панели, все запросы на

сервер в панели являются асинхронными.

Для каждого пункта панели создаются привилегии на просмотр, создание, изменение, удаление и восстановление. Эти привилегии находятся в конфигурационном файле. Затем они используются в политиках проекта, которые проверяют наличие привилегий. Например, вот привилегии для списка пользователей:

```
class AdminUserPolicy
{
    use HandlesAuthorization;

    public function index(User $user)
    {
        return $user->hasPrivilege('admin.user.index');
    }

    public function show(User $user)
    {
        return $user->hasPrivilege('admin.user.show');
    }

    public function store(User $user)
    {
        return $user->hasPrivilege('admin.user.store');
    }

    public function update(User $user)
    {
        return $user->hasPrivilege('admin.user.update');
    }

    public function destroy(User $user)
    {
        return $user->hasPrivilege('admin.user.destroy');
    }

    public function restore(User $user)
    {
        return $user->hasPrivilege('admin.user.restore');
    }
}
```

Эти функции проверки привилегий потом используются в контроллерах

для проверки. Ниже представлен код функции просмотра списка пользователей в административной панели:

```
public function index(Request $request)
{
    $this->authorize('index', AdminFeedbackPolicy::class);

    $feedbackGetter = new FeedbackGetter();

    return $feedbackGetter->search($request->input('search'))
        ->withTrashed($request->input('withTrashed'))
        ->orderBy($request->input('order_by'), $request-
        >input('order_dir'))
        ->paginate();
}
```

Из кода выше видно, что функция `authorize` отвечает за проверку привилегий.

Для получения данных из БД используются геттеры, а для сохранения в БД используются сеттеры. Для каждого типа данных используются разные геттеры и сеттеры.

Административная панель находится в папке `resources/assets/admin`, для разделения пользовательской и административной части приложения. Для каждого пункта меню реализованы 2 страницы: список данных и создание/редактирование определённых данных. Для примера представлен код страницы списка пользователей:

```
<template>
    <el-main v-if="appData">
        <el-breadcrumb separator-class="el-icon-arrow-right"
            class="margin-bottom-20">
            <el-breadcrumb-item>Панель управления</el-breadcrumb-
            item>
            <el-breadcrumb-item>Пользователи</el-breadcrumb-item>
        </el-breadcrumb>

        <el-row class="margin-bottom-20">
            <el-col :span="12">
                <router-link :to="{name: routeName + '.create'}"
                v-if="$store.getters.hasPrivilege('admin.user.store')">
                    <el-button type="primary">Добавить</el-button>
            </el-col>
        </el-row>
    </el-main>
</template>
```

```

        </router-link>
    </el-col>
</el-row>

<el-table
  @sort-change="index(null, $event)"
  :data="appData.data"
  border
  :default-sort="defaultSort"
  :class="'search margin-bottom-20'"
  style="width: 100%">
  <el-table-column>
    <el-table-column
      prop="id"
      label="№"
      sortable="custom"
      width="100">
    </el-table-column>
  </el-table-column>
  <el-table-column>
    <template slot="header" slot-scope="scope">
      <el-input v-model="search['users.name']"
        clearable
        placeholder="Поиск по Имени"
        size="mini"
        @keyup.native.enter="index(1)"/>
    </template>
    <el-table-column
      prop="name"
      label="Имя"
      sortable="custom">
    </el-table-column>
  </el-table-column>
  <el-table-column>
    <template slot="header" slot-scope="scope">
      <el-input v-model="search['users.surname']"
        clearable
        placeholder="Поиск по Фамилии"
        size="mini"
        @keyup.native.enter="index(1)"/>
    </template>
    <el-table-column
      prop="surname"
      label="Фамилия"
      sortable="custom">

```

```

        </el-table-column>
    </el-table-column>
    <el-table-column>
        <template slot="header" slot-scope="scope">
            <el-input v-model="search['users.email']"
                clearable
                placeholder="Поиск по email"
                size="mini"
                @keyup.native.enter="index(1)"/>
        </template>
        <el-table-column
            prop="email"
            label="Email"
            sortable="custom">
        </el-table-column>
    </el-table-column>
    <el-table-column align="center">
        <template slot="header">
            <el-button size="mini"
@click="clearSearch()">Очистить</el-button>
        </template>
        <el-table-column align="center" width="110">
            <template slot-scope="scope">
                <router-link :to="{name: routeName +
'.edit', params: {id: scope.row.id}}"
                    v-
if="$store.getters.hasPrivilege('admin.user.update')">
                    <el-button size="mini" type="primary"
icon="el-icon-edit" circle></el-button>
                </router-link>
                <el-button size="mini" type="danger"
icon="el-icon-delete" circle @click="destroy(scope.row)"
                    v-if="!scope.row.deleted_at &&
$store.getters.hasPrivilege('admin.user.destroy')"></el-button>
                <el-button size="mini" icon="el-icon-
refresh" circle @click="restore(scope.row)"
                    v-if="scope.row.deleted_at &&
$store.getters.hasPrivilege('admin.user.restore')"></el-button>
            </template>
        </el-table-column>
    </el-table-column>
</el-table>

<el-pagination
    background

```

```

        layout="prev, pager, next"
        :total="appData.total"
        :page-size="appData.per_page"
        :current-page="appData.current_page"
        @current-change="index($event) ">
    </el-pagination>
</el-main>
</template>

<script>
import listMixin from "../../mixins/list";
import WithTrashed from "../../layout/components/WithTrashed";

export default {
  mixins: [listMixin],
  data: () => ({
    storeName: 'user',
    apiUrl: '/api/admin/user'
  }),
  components: {
    WithTrashed
  },
}
</script>

```

Это код компонента Vue [4] – JS фреймворка, который разделяет код на 3 части: HTML, CSS и JS.

В этом коде описан показ пользователей через асинхронный запрос. Тут также есть проверка на наличие привилегий.

Ниже представлен код изменения пользователя:

```

<template>
  <el-main v-if="currentItem">
    <el-breadcrumb separator-class="el-icon-arrow-right"
class="margin-bottom-20">
      <el-breadcrumb-item>Панель управления</el-breadcrumb-
item>
      <el-breadcrumb-item :to="{name:
routeName}">Пользователи</el-breadcrumb-item>
      <el-breadcrumb-item>{{ currentItem.id }}
'Редактировать' : 'Создать' }} пользователя</el-breadcrumb-item>
    </el-breadcrumb>

    <el-form ref="form" :model="currentItem"

```

```

@submit.prevent.native="submit()">
  <el-row :gutter="20">
    <el-col :md="6">
      <el-form-item label="ФИО *">
        <el-input v-model="currentItem.name"
required></el-input>
      </el-form-item>
    </el-col>
    <el-col :md="6">
      <el-form-item label="Email *">
        <el-input type="email" v-
model="currentItem.email" required></el-input>
      </el-form-item>
    </el-col>
    <el-col :md="6">
      <el-form-item label="Телефон *">
        <el-input v-model="currentItem.phone" v-
mask="' +7 ### ### #' " placeholder="+79999999999"
required></el-input>
      </el-form-item>
    </el-col>
    <el-col :md="6">
      <el-form-item :label="'Пароль ' +
(currentItem.id ? ' ' : ' *')">
        <el-input type="password"
:required="!currentItem.id"
v-
model="currentItem.new_password">
        </el-input>
      </el-form-item>
    </el-col>
    <el-col :lg="24" :xl="12">
      <el-form-item>
        <el-transfer v-
model="currentItem.new_roles"

:data="$store.getters['data/role']"

:props="{key: 'id', label:
'description'}"

:titles="['Все роли',
'Выбранные роли']"

:filter-
placeholder="'Пусто'">

      </el-transfer>

```

```

        </el-form-item>
      </el-col>
      <el-col :md="24">
        <el-form-item>
          <el-button type="primary" native-
type="submit" :loading="loading"
                                v-if="!currentItem.id    &&
$store.getters.hasPrivilege('admin.user.store') ">
                                Создать
          </el-button>
          <el-button type="primary" native-
type="submit" :loading="loading"
                                v-if="currentItem.id      &&
$store.getters.hasPrivilege('admin.user.update') ">
                                Сохранить
          </el-button>
          <router-link :to="{name: routeName}">
            <el-button>Отмена</el-button>
          </router-link>
        </el-form-item>
      </el-col>
    </el-row>
  </el-form>
</el-main>
</template>

```

```

<script>
  import editMixin from "../../mixins/edit";
  import {mask} from "vue-the-mask";

  const forEach = require('lodash/forEach');

  export default {
    mixins: [editMixin],
    directives: {mask},
    data: () => ({
      storeName: 'user',
      apiUrl: '/api/admin/user'
    }),
    watch: {
      currentItem() {
        if (!this.currentItem.new_roles) {
          this.$set(this.currentItem, 'new_roles', []);

          forEach(this.currentItem.roles, (item) => {

```

```

        this.currentItem.new_roles.push(item.id);
    });
}
    },
    mounted() {
        this.$store.dispatch('data/get', {path: '/role/all',
type: 'role'})
        .catch((error) => {
            this.$appNotify.error(error.response.data);
        });
    }
}
</script>

```

В этом коде видно, что администратор может поменять имя, телефон, пароль и роль данного пользователя.

Все данные, которые необходимы всем страницам административной панели находятся в хранилище данных, называемым Vuex [5], он позволяет компоненту Vue получать необходимые данные без повторного запроса при переходе на другую страницу панели.

Вид административной панели представлен на рисунке 2.

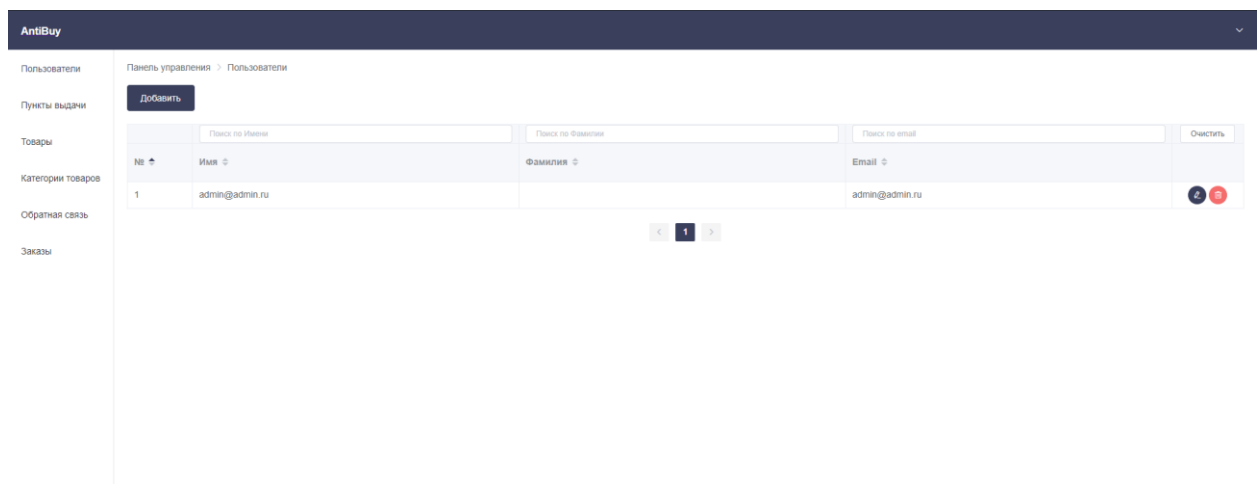


Рисунок 2 – Административная панель

3.5 Разработка страницы показа товаров

Была сделана таблица «products» и «product_categories» в БД. Затем был создан контроллер Front/HomeController.php, который отвечает за вызов страницы home по маршруту «/». Сама страница home, представляет собой компонент Vue, в котором находятся фильтры и сами товары, они так же разделены на отдельные компоненты. Так как контроллер отвечает только за вызов страницы, то есть за запросы, которые происходят при загрузке страницы, то для получения товаров нужен другой контроллер в папке API, который отвечает за асинхронные запросы. Ниже представлена функция index контроллера ProductController.php, которая передает данные о продуктах компоненту Vue:

```
public function index(Request $request)
{
    $productGetter = (new ProductGetter())
        ->getSearch($request->input('filter'))
        ->orderBy($request->input('filter.sort'));

    $products = $productGetter->get();

    foreach ($products as &$product) {
        (new ProductService($product))
            ->withPhotos()
            ->withMainImage()
            ->withUrl()
            ->withOrder()
            ->get();
    }

    return $products;
}
```

Из кода видно, что данные о продуктах передаются из геттера продуктов. Также здесь используется сервис товаров, который манипулирует с данными для получения дополнительной информации, например ссылки на товар или принадлежащих товару фотографий. Код сервиса представлен ниже:

```

class ProductService
{
    protected $instance = null;

    public function __construct($instance)
    {
        $this->instance = $instance;
    }

    public function withPhotos()
    {
        $photos = $this->instance->getMedia('photos');

        $data = [];

        foreach ($photos as $photo) {
            $data[] = [
                'id' => $photo->id,
                'url' => $photo->getUrl(),
                'name' => $photo->getCustomProperty('name'),
                'desc' => $photo->getCustomProperty('desc')
            ];
        }

        $this->instance->photos = collect($data);

        return $this;
    }

    public function withPhotosUrls()
    {
        $mainImage = $this->instance->getMedia('main_image')->last();
        $photos = $this->instance->getMedia('photos');

        $data = [];

        if ($mainImage) {
            array_push($data, $mainImage->getUrl());
        }

        foreach ($photos as $photo) {
            array_push($data, $photo->getUrl());
        }
    }
}

```

```

        $this->instance->photos = collect($data);

        return $this;
    }

    public function withMainImage()
    {
        $mainImage      =      $this->instance->getMedia('main_image')->last();

        if ($mainImage) {
            $this->instance->main_image = [
                'id' => $mainImage->id,
                'url' => $mainImage->getUrl(),
                'name' => $mainImage->getCustomProperty('name'),
                'desc' => $mainImage->getCustomProperty('desc')
            ];
        }

        return $this;
    }

    public function withAdminMainImage()
    {
        $this->withMainImage();

        $data[] = $this->instance->main_image;

        $this->instance->main_image = collect($data);

        return $this;
    }

    public function withUrl()
    {
        $this->instance->url = route('product', $this->instance->id);

        return $this;
    }

    public function withOrder()
    {
        $this->instance->order = Order::where('product_id', $this->instance->id)

```

```

        ->where('order_status_id',                                '!=',
OrderStatus::CANCELED_ADMIN)
        ->where('order_status_id',                                '!=',
OrderStatus::CANCELED_USER)
        ->count();

        return $this;
    }

    public function get()
    {
        return $this->instance;
    }
}

```

Так как все вызовы к БД используют функции фреймворка Laravel, то отсутствует уязвимость SQL-инъекций. Также была разработана страница товаров в административной панели, которая по функционалу похожа на страницу пользователей.

3.6 Разработка детальной страницы товара

Детальная страница товара работает также как и показ всех товаров, только вместо асинхронного запроса использует запрос при загрузке страницы.

Ниже представлен код контроллера Front/ProductController.php:

```

class ProductController extends Controller
{
    public function show($id)
    {
        $product = Product::with('productCategories')
            ->findOrFail($id);

        $product = (new ProductService($product))
            ->withPhotosUrls()
            ->withUrl()
            ->withOrder()
            ->get();

        ToJs::put([
            'product' => $product,

```

```

        'user' => \Auth::user(),
    ]);

    return view('front.sections.product.index');
} }

```

Тут используется функция ToJs для передачи данных в клиентскую часть приложения.

3.7 Разработка заказа

Была сделана таблица «orders» и «order_statuses» в БД. Заказ товара был встроен в детальную страницу товара. Была разработана страница заказов в административной панели, которая по функционалу похожа на предыдущие страницы, кроме того, что администратор не может создавать или изменять заказ помимо его статуса. Вид заказа в административной панели представлен на рисунке 3.

Панель управления > **Заказы** > Изменить заказ

ID заказчика	ФИО заказчика	Телефон заказчика
<input type="text" value="1"/>	<input type="text" value="admin@admin.ru"/>	<input type="text"/>
ID товара	Название товара	Цена товара
<input type="text" value="1"/>	<input type="text" value="Полное собрание сочинений Михаила Васи"/>	<input type="text" value="41000"/>
ID пункта выдачи	Название пункта выдачи	
<input type="text" value="1"/>	<input type="text" value="Пенза"/>	

Статус:

- ☒ Новый
- ☐ В работе
- ☐ Отменён пользователем
- ☐ Отменён администратором
- ☐ Доставлен в пункт выдачи
- ☐ Получен пользователем и оплачен

Рисунок 3 – Страница изменения заказа в административной панели

Код заказа товара представлен ниже:

```
class OrderController extends Controller
```

```

{
    public function create(Request $request)
    {
        DB::beginTransaction();

        $orderSaver = new OrderSaver();

        $orderSaver->setUser($request->input('user_id'))
            ->setOffice($request->input('office_id'))
            ->setProduct($request->input('product_id'))
            ->setStatus(OrderStatus::NEW_ORDER)
            ->save();

        DB::commit();

        return $orderSaver->getInstance();
    }
}

```

По коду видно, что используется сеттер идентификаторов пользователя, пункта выдачи, товара и устанавливается статус для заказа.

Заказать товар может только авторизованный пользователь. Если товар заказан, то он исчезает из списка товаров на сайте и, если перейти на детальную страницу товара, будет уведомление о том, что товар уже заказан.

На рисунке 4 представлен процесс заказ для пользователя.

листов гравюр.
Том из первого научного издания собрания сочинений Ломоносова. Полный комплект гравюр.
СК XVIII № 3733.

Выберите пункт доставки:

Пенза

проспект Победы, 126

12:00-18:00 каждый день

92754544343, 543589345934

автобус 54

Пенза-1

жилой комплекс Новелла, с3

12:00-18:00 каждый день

92754544343, 543589345934

автобус

Заказать

Рисунок 4 – Заказ товара

Код клиентской части детальной страницы и заказа представлен ниже:

```
<template>
  <div class="product" v-if="product">
    <div class="product__top">
      <div class="product__name">{{ product.title }}</div>
      <div class="product__tags">
        <template v-for="category in
product.product_categories">{{ category.title }}</template>
      </div>
      <div class="product__info-item">
        <span class="product__info-name">Цена</span>
        <span class="product__price">{{
Number(product.price).toLocaleString('ru') }} ₸</span>
      </div>
    </div>
    <vue-gallery-slideshow :images="product.photos"
:index="image" @close="image = null"></vue-gallery-slideshow>
    <div class="product__images">
      <div class="product__main-img" :style="'background-
image: url('+ product.photos[0] + ');' " @click="image = 0" />
      <div class="product__images-box n-scroll"
:style="windowWidth - 20 > 82 * product.photos.slice(1).length &&
windowWidth < 1170 ? 'justify-content: center;' : ''">
        <div class="product__img" v-for="(item, i) in
product.photos.slice(1)" :key="i" @click="image = (i + 1)"
:style="'background-image: url('+ item + ');' " />
        <button class="btn product__scroll"
@click="imagesScroll()" v-if="(windowWidth >= 1170 &&
product.photos.slice(1).length > 2) || (windowWidth - 20 < 82 *
product.photos.slice(1).length && windowWidth < 1170)" />
      </div>
    </div>
    <div class="product__description" v-
html="textFormat(product.description)"></div>
    <button class="btn btn--main btn--middle"
@click="showOrder()" v-if="!show_order" :disabled="user.length ===
0 || product.order > 0">
      Заказать
      <i class="el-icon-arrow-down" style="margin-left:
10px"></i>
    </button>
    <div class="product__hint" v-if="product.order > 0">
      Товар уже куплен
    </div>
```

```

<div class="product__hint" v-if="user.length === 0">
    Войдите или зарегистрируйтесь для заказа товара
</div>
<div v-loading="loading" class="product__order" v-else>
    <template v-if="!order_create">
        <template v-if="offices.length > 0">
            <div class="product__order-title">Выберите
пункт доставки:</div>
            <div class="product__order-inner">
                <div class="product__order-items">
                    <div class="product__order-item"
                        @click="changeOffice(item)"
                        :class="{ 'product__order-item--
active': office === item.id}"
                        v-for="item in offices"
                        :key="item.id">
                        <div class="product__order-item-
name">{{ item.branch_name }}</div>
                        <div class="product__order-item-
value">{{ item.address }}</div>
                        <div class="product__order-item-
value">{{ item.working_hours }}</div>
                        <div class="product__order-item-
value">{{ item.contact_phones }}</div>
                        <div class="product__order-item-
value">{{ item.how_to_get }}</div>
                    </div>
                    <div>
                        <yandex-map class="product__order-map"
                            :coords="coords"
                            :zoom="14">
                            <ymap-marker v-for="office in offices"
                                :key="office.id"
                                :coords="[office.latitude, office.longitude]"
                                :marker-id="office.id"
                                :hint-
content="office.branch_name"/>
                        </yandex-map>
                    </div>
                    <button class="btn btn--main btn--middle"
                        @click="createOrder()">Заказать</button>
                </template>
            </template>
            <div class="product__order-text" v-else>Заказ успешно

```

```
создан. Вы можете перейти в историю заказов для его
просмотра</div>
</div>
</div>
</template>
```

```
<script>
import VueGallerySlideshow from 'vue-gallery-slideshow';
import {mapState} from 'vuex';
import {store} from "../../store/product";
import style from "../../mixins/style";
import ApiService from "../../js/modules/ApiService";

export default {
  name: "product",
  store,
  mixins: [style],
  components: {
    VueGallerySlideshow
  },
  computed: {
    ...mapState([
      'product', 'user'
    ]),
  },
  data: () => ({
    image: null,
    show_order: false,
    offices: [],
    office: null,
    coords: [],
    loading: false,
    order_create: false,
  })),

  methods: {
    imagesScroll() {
      if (this.windowWidth >= 1170) {
        document.querySelector('.product__images-
box').scrollTop = document.querySelector('.product__images-
box').scrollHeight;
      } else {
        document.querySelector('.product__images-
box').scrollLeft = document.querySelector('.product__images-
box').scrollWidth;
      }
    }
  }
}
```

```

        }
    },

    showOrder() {
        this.show_order = true;
        new ApiService().get('/all-offices')
            .then((response) => {
                this.offices = response.data;
                this.coords = [this.offices[0].latitude,
this.offices[0].longitude];
                this.office = this.offices[0].id;
            })
            .catch((error) => {
                this.$appNotify.error(error.response.data);
            })
    },

    changeOffice(office) {
        this.office = office.id;
        this.coords = [office.latitude, office.longitude];
    },

    createOrder() {
        this.loading = true;
        let order = {
            user_id: this.user.id,
            product_id: this.product.id,
            office_id: this.office
        };
        new ApiService().post('/create-order', order)
            .then((response) => {
                this.loading = false;
                this.order_create = true;
            })
            .catch((error) => {
                this.loading = false;
                this.$appNotify.error(error.response.data);
            })
    }
}
}
</script>

```

3.8 Разработка страницы предыдущих заказов

Страница заказов использует данные таблицы заказов из БД. На странице заказов можно отменить заказ. Код функций показа заказов и отмены представлен ниже:

```
public function index()
{
    $orders = Order::with(Order::$allRelations)-
>where('user_id', \Auth::id())->get();

    foreach ($orders as &$order) {
        (new ProductService($order->product))
            ->withUrl()
            ->get();
    }

    return $orders;
}

public function cancel(Request $request)
{
    $order = Order::findOrFail($request->input('id'));

    DB::beginTransaction();

    $orderSaver = new OrderSaver($order);

    $orderSaver
        ->setStatus(OrderStatus::CANCELED_USER)
        ->save();

    DB::commit();

    return $orderSaver->getInstance();
}
```

На рисунке 5 представлена страница заказов пользователя.

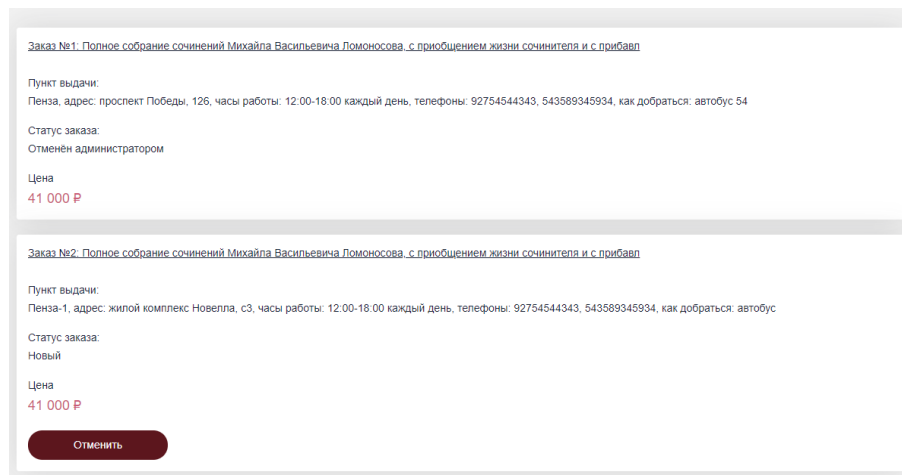


Рисунок 5 – Заказы пользователя

Код клиентской части страницы заказов представлен ниже:

```
<template>
  <div class="history container" v-if="orders">
    <div class="history__items" v-infinite-scroll="load"
infinite-scroll-disabled="loading">
      <div class="history__item" v-for="order in orders">
        <a
class="history__name">Заказ №{{ order.id }}: {{
order.product.title }}</a>
        <div class="history__info">
          <div class="history__info-item">
            <div class="history__info-name">Пункт
выдачи:</div>
            <div class="history__info-value">
              {{ order.office.branch_name }},
              адрес: {{ order.office.address }},
              часы работы: {{
order.office.working_hours }},
              телефоны: {{
order.office.contact_phones }},
              как добраться: {{
order.office.how_to_get }}
            </div>
          </div>
          <div class="history__info-item">
            <div class="history__info-name">Статус
заказа:</div>
            <div class="history__info-value">{{
order.order_status.title }}</div>
          </div>
          <div class="history__info-item">
```

```

        <div class="history__info-name">Цена</div>
        <div class="history__price">{{
Number(order.product.price).toLocaleString('ru')}} ₸</div>
        </div>
        <button class="btn btn--middle btn--main"
@click="cancelOrder(order.id)" v-if="order.order_status_id != 3 &&
order.order_status_id != 4 && order.order_status_id !=
6">ОТМЕНИТЬ</button>
        </div>
    </div>
</div>
<div v-if="loading" class="history__loading-box">
    <i class="history__loading el-icon-loading"></i>
</div>
</div>
</template>

```

```

<script>
import {mapState, mapActions, mapMutations} from 'vuex';
import {store} from "../../store/history";
import style from "../../mixins/style";
import ApiService from "../../js/modules/ApiService";

```

```

export default {
  name: "History",
  store,
  mixins: [style],
  data: () => ({
    count: 8,
    loading: false,
    data: [],
  }),
  computed: {
    ...mapState([
      'orders',
    ]),
  },
  methods: {
    ...mapMutations([
      'setOrders'
    ]),
    ...mapActions([
      'getOrders'
    ]),
    load() {

```

```

        if (this.count < this.data.length) {
            this.loading = true
            setTimeout(() => {
                this.count += 16
                this.loading = false
            }, 2000)
        }
    },

    cancelOrder(id) {
        new ApiService().post('/cancel-order', {id: id})
        .then((response) => {
            this.getOrders();
        });
    },

    },

    mounted() {
        this.getOrders();
    }
}
</script>

```

3.9 Разработка страницы обратной связи

Была создана таблица «comments» в БД и разработана форма отправки сообщения. Сообщения могут опубликовать только авторизованные пользователи. Другие не смогут попасть на эту страницу.

Код функции отправки сообщения представлен ниже:

```

class FeedbackController extends Controller
{
    public function send(Request $request)
    {
        DB::beginTransaction();

        $feedbackSaver = new FeedbackSaver();

        $feedbackSaver->fill($request->all())
            ->setFeedbackUserId($request->input('user_id'))
            ->save();

        DB::commit();
    }
}

```

```

        return $feedbackSaver->getInstance();
    }
}

```

Код клиентской части страницы отправки сообщения представлен ниже:

```

<template>
  <form class="feedback" @submit.prevent="sendFeedback">
    <h2 class="page__title title title--m">Обратная связь</h2>
    <template v-if="!send">
      <div class="feedback__description">
        Здесь Вы можете направить своё обращение
      </div>
      <label class="form__item">
        <span class="form__label">Название
сообщения</span>
        <el-input class="input" v-model="feedback.title"
required/>
      </label>
      <label class="form__item">
        <span class="form__label">Текст сообщения</span>
        <el-input type="textarea" class="input"
:rows="windowWidth > 992 ? 4 : 6" v-model="feedback.description"
required/>
      </label>
      <div class="form__item form__item--center">
        <button class="btn btn--middle btn--main
feedback__button"
          type="submit">
          Отправить
        </button>
      </div>
    </template>
    <div class="feedback__description" v-else>
      Ваше обращение успешно отправлено!
    </div>
  </form>
</template>

<script>
  import {mask} from "vue-the-mask";
  import ApiService from "../../../../../js/modules/ApiService";

  export default {
    name: "Feedback",

```

```

    directives: {mask},
    props: ['user'],
    data: () => ({
      feedback: {
        title: '',
        description: '',
        user_id: ''
      },
      send: false,
    }),
    methods: {
      sendFeedback() {
        this.$set(this.feedback, 'user_id', this.user.id);
        new ApiService().post('/send-feedback',
this.feedback)
          .then((response) => {
            this.send = true;
          });
      },
    }
  }
}
</script>

```

3.10 Разработка страницы показа отзывов

Для страницы отзывов была использована таблица «comments» в БД. Для получения данных были использованы геттеры.

Код функции отзывов представлен ниже:

```

class CommentController extends Controller
{
  public function index()
  {
    $commentsGetter = new CommentGetter();

    ToJs::put([
      'comments' => $commentsGetter->get(),
    ]);

    return view('front.sections.comments.index');
  }
}

```

Код клиентской части страницы отзывов представлен ниже:

```

<template>
  <div class="comments">
    <h2 class="page__title title title--m">Отзывы</h2>
    <div class="comments__items">
      <div class="comments__item" v-for="(comment, i) in
comments">
        <span class="comments__name">
          {{ comment.title }} <span
class="comments__city" v-if="comment.name">{{ comment.name
}}</span>
        </span>
        <span class="comments__quote">"</span>
        <div class="comments__text comments__text--small">
          <span>{{ comment.description }}</span>
        </div>
        <button class="btn comments__full"
@click="openComment(comment)" v-if="comment.description.length >
210">Показать полностью...</button>
      </div>
    </div>
    <full-comment></full-comment>
  </div>
</template>

<script>
import {store} from "../../store/comments";
import {mapState, mapActions} from "vuex";
import FullComment from "../modules/FullComment";

export default {
  name: "Comments",
  components: {FullComment},
  store,
  computed: {
    ...mapState([
      'comments',
      'showComment'
    ]),
  },
  methods: {
    ...mapActions([
      'openComment',
    ]),
  },
}

```

</script>

3.11 Развертывание сайта на сервере и его настройка

Проект был клонирован с удаленного репозитория и помещен в папку /var/www/diplom. Затем были установлены зависимости командами composer install и npm install, установлен файл конфигурации .env, и сделаны миграции БД.

На тестовом сервере стоит сервер Nginx. В папке /etc/nginx/sites-available был создан файл виртуального хоста diplom. Код файла представлен ниже:

```
server {
    listen 80;
    server_name diplom.rosoperator.com;
    root    /var/www/diplom/public;
    index   index.html;

    client_max_body_size 256m;

    location / {
        try_files $uri $uri/ /index.php?$query_string;
    }

    location ~* ^.+.(js|css|png|jpg|jpeg|gif|ico)$ {
        access_log      off;
        expires          max;
    }

    location ~ \.php$ {
        # fastcgi_split_path_info ^(.+\.(php))(.*)$;
        #fastcgi_pass     127.0.0.1:9000;
        fastcgi_pass unix:/run/php/php7.4-fpm.sock;
        fastcgi_index    index.php;

        include fastcgi_params;

        fastcgi_param    SCRIPT_FILENAME
$document_root$fastcgi_script_name;
        try_files $uri =404;
        fastcgi_read_timeout 600;
    }
```

```

        location = /favicon.ico {
            log_not_found off;
            access_log off;
        }

        location = /robots.txt {
            allow all;
            log_not_found off;
            access_log off;
        }

        ## Disable viewing .htaccess & .htpassword
        location ~ /\.ht {
            deny  all;
        }
    }
}

```

Затем была создана ссылка на этот файл в папке `etc/nginx/sites-enabled` и перезагружен сервер. На предприятии был получен домен `diplom.rosoperator.com`. Пока проект работал по протоколу HTTP.

3.12 Подключение SSL для HTTPS

Let's Encrypt [6] — это центр сертификации, позволяющий легко получать и устанавливать бесплатные сертификаты TLS/SSL, что позволяет использовать на веб-серверах шифрованный трафик HTTPS. Это упрощает процесс посредством предоставления программного клиента Certbot, который пытается автоматизировать большинство необходимых шагов (или все шаги).

Certbot предоставляет широкий выбор способов получения сертификатов SSL с помощью плагинов: Плагин Nginx изменит конфигурацию Nginx и перезагрузит ее, когда это потребуется. Для использования этого плагина была введена следующая команда: `sudo certbot --nginx -d diplom.rosoperator.com`.

Эта команда запускает certbot с плагином `--nginx`, используя опцию `-d` для указания доменных имен, для которых мы хотим использовать сертификат.

Certbot автоматически исправил конфигурационный файл для перенаправления с протокола HTTP на HTTPS.

Код файла виртуального хоста представлен ниже:

```
server {
    server_name diplom.rosoperator.com;
    root    /var/www/diplom/public;
    index   index.html;

    client_max_body_size 256m;

    location / {
        try_files $uri /index.php?$query_string;
    }

    location ~* ^.+.(js|css|png|jpg|jpeg|gif|ico)$ {
        access_log      off;
        expires          max;
    }

    location ~ \.php$ {
        # fastcgi_split_path_info ^(.+\.(php))(.*)$;
        #fastcgi_pass    127.0.0.1:9000;
        fastcgi_pass unix:/run/php/php7.4-fpm.sock;
        fastcgi_index   index.php;

        include fastcgi_params;

        fastcgi_param   SCRIPT_FILENAME
$document_root$fastcgi_script_name;
        try_files $uri =404;
        fastcgi_read_timeout 600;
    }

    location = /favicon.ico {
        log_not_found off;
        access_log off;
    }

    location = /robots.txt {
        allow all;
        log_not_found off;
        access_log off;
    }
}
```

```

        ## Disable viewing .htaccess & .htpassword
        location ~ /\.ht {
            deny  all;
        }

        listen 443 ssl; # managed by Certbot
        ssl_certificate
        /etc/letsencrypt/live/diplom.rosoperator.com/fullchain.pem;          #
        managed by Certbot
        ssl_certificate_key
        /etc/letsencrypt/live/diplom.rosoperator.com/privkey.pem;          #
        managed by Certbot
        include /etc/letsencrypt/options-ssl-nginx.conf; # managed by
        Certbot
        ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by
        Certbot

    }

    server {
        if ($host = diplom.rosoperator.com) {
            return 301 https://$host$request_uri;
        } # managed by Certbot

        listen 80;
        server_name diplom.rosoperator.com;
        return 404; # managed by Certbot
    }

```

Затем был перезагружен сервер и сайт начал работать на протоколе HTTPS.

4 Функциональное тестирование разработанного веб-сайта от рассмотренных угроз

4.1 Инъекционная атака

Так как в приложении используются функции Laravel, то переданные параметры точно попадают в оператор запроса и не могут за него выйти.

На сайте есть URL, в который передается идентификатор товара: <https://diplom.rosoperator.com/product/1>, где 1 – идентификатор товара.

Для проверки было использована простая атака, где к 1 было добавлено `order by 1000`, результат представлен на рисунке 5.

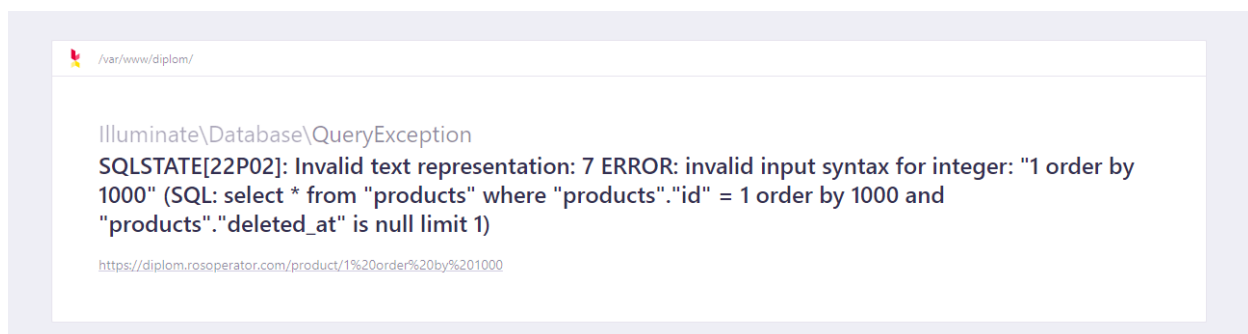


Рисунок 5 - Результат атаки

Из ошибки видно, что `order by 1000` попало в оператор полностью и не вышло за его пределы.

4.2 Подбор пароля

В контроллере управления авторизацией стоит счетчик неправильных входов в аккаунт. Было проверено сколько раз можно вводить неправильный пароль до блокировки. На 6 раз ввода приложение выдало ошибку о большом количестве попыток входа. Это представлено на рисунке 6.

АВТОРИЗАЦИЯ

[Зарегистрироваться](#)

E-mail

admin@admin.ru

Too many login attempts. Please try again in 53 seconds.

Введите пароль

Введите пароль

☐

Запомнить меня

Войти

Рисунок 6 – Ошибка входа

4.3 Перехват данных клиента и сервера при обмене

Для проверки этой атаки была запущена утилита Wireshark. С помощью этой утилиты была произведена попытка перехвата трафика от клиента до сервера. Результат представлен на рисунке 7.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.1.101	5.200.40.155	TCP	1514	50421 → 443 [ACK] Seq=
2	0.000018	192.168.1.101	5.200.40.155	TLSv1.2	334	Application Data
6	0.038411	192.168.1.101	5.200.40.155	TCP	66	50421 → 443 [ACK] Seq=
17	0.512757	192.168.1.101	5.200.40.155	TCP	1514	50421 → 443 [ACK] Seq=
18	0.512787	192.168.1.101	5.200.40.155	TLSv1.2	232	Application Data
21	0.585069	192.168.1.101	5.200.40.155	TCP	66	50421 → 443 [ACK] Seq=
273	37.070824	192.168.1.101	5.200.40.155	TCP	1514	50421 → 443 [ACK] Seq=
274	37.070842	192.168.1.101	5.200.40.155	TLSv1.2	444	Application Data
275	37.070882	192.168.1.101	5.200.40.155	TLSv1.2	180	Application Data
280	37.184349	192.168.1.101	5.200.40.155	TCP	66	50421 → 443 [ACK] Seq=
281	37.196195	192.168.1.101	5.200.40.155	TCP	1514	50421 → 443 [ACK] Seq=
282	37.196214	192.168.1.101	5.200.40.155	TLSv1.2	329	Application Data
286	37.235012	192.168.1.101	5.200.40.155	TCP	66	50421 → 443 [ACK] Seq=
289	37.533907	192.168.1.101	5.200.40.155	TCP	1514	50421 → 443 [ACK] Seq=
290	37.533924	192.168.1.101	5.200.40.155	TLSv1.2	509	Application Data
291	37.534520	192.168.1.101	5.200.40.155	TCP	74	50466 → 443 [SYN] Seq=
295	37.557093	192.168.1.101	5.200.40.155	TCP	66	50466 → 443 [ACK] Seq=
296	37.557401	192.168.1.101	5.200.40.155	TLSv1.2	589	Client Hello
298	37.578093	192.168.1.101	5.200.40.155	TCP	1514	50421 → 443 [ACK] Seq=
299	37.578112	192.168.1.101	5.200.40.155	TLSv1.2	602	Application Data
300	37.578153	192.168.1.101	5.200.40.155	TLSv1.2	139	Application Data
303	37.579988	192.168.1.101	5.200.40.155	TLSv1.2	117	Change Cipher Spec, En

Рисунок 7 – Перехваченные пакеты

Из рисунка видно, что для передачи данных используется протокол TLSv1.2, что значит трафик передается в зашифрованном виде.

4.4 Межсайтовый скриптинг (XSS)

Для тестирования защиты от этой атаки была использована страница отзывов. В ней текст, который идет с сервера, содержимое экранируется и поэтому HTML теги появляются как простой текст. Результат атаки представлен на рисунке 8.



Проверка XSS admin@admin.ru
“
<script>alert(123)</script>

Рисунок 8 – Проверка атаки XSS

Как видно из рисунка HTML теги появились, значит уязвимость от атак XSS отсутствует.

4.5 Межсайтовая подделка запроса

Для проверки этой атаки была создана HTML форма на локальном сервере. Код формы представлен ниже:

```
<form      action="https://diplom.rosoperator.com/api/send-feedback"
method="post">
    <input          type="hidden"          name="description"
value="dasdasdasdasdasda xss">
    <input type="hidden" name="title" value="dasdasdasdasda">
    <input type="hidden" name="user_id" value="1">
    <input type="submit" value="submit">
</form>
```

Была произведена попытка отправить данные с другого сайта. В результате сайт diplom.rosoperator.com перебросил на свою страницу входа.

5 Вредные психофизиологические факторы, влияющие на нервную систему специалиста по защите информации. Методы и средства физической культуры, снижающие их воздействие

К вредным производственным факторам, обладающим свойствами психофизиологического воздействия на нервную систему специалиста по защите информации, относятся нервно-психические перегрузки, связанные с напряженностью трудового процесса.

Согласно ГОСТ 12.003–2015 [7], нервно-психические перегрузки подразделяют на:

- умственное перенапряжение, в том числе вызванное информационной нагрузкой;
- перенапряжение анализаторов, в том числе вызванное информационной нагрузкой;
- монотонность труда, вызывающая монотонию;
- эмоциональные перегрузки.

Основная деятельность специалистов по защите информации связана с умственным напряжением, эмоционально-умственным напряжением, творческим и умственным трудом.

Умственная деятельность — это деятельность, прежде всего, центральной нервной системы, ее высшего отдела — коры человеческого мозга.

При умственной работе, как и при физической, изменяются обменные процессы, происходит сужение сосудов конечностей и расширение сосудов внутренних органов, пульс изменяется незначительно, потребление кислорода мозгом увеличивается в 15–20 раз при умственной работе, по сравнению с физической.

Кроме того, в случаях, когда умственная работа сопровождается нервноэмоциональным напряжением, то возможны значительные изменения кровяного давления, пульса, повышение уровня сахара в крови [8].

Самым естественным и простым вариантом устранения умственного переутомления является сон. Во время сна упорядочивается и фильтруется информация, полученная в процессе дневной умственной деятельности. Кроме того, клетки головного мозга очищаются от нейротоксинов. Способность к концентрации внимания и запоминанию большого объема информации тоже зависят от качества и продолжительности сна. Считается, что нормальная продолжительность сна составляет около 8 часов в сутки, но, если сотрудник страдает от умственной усталости, ему понадобится увеличить его продолжительность до 10÷12 часов. Для лечения и профилактики умственного перенапряжения применяются также медитация, экотерапия (созерцание красот природы или природных явлений), «информационное голодание» или «разгрузочные дни» для ума [9].

Центральная нервная система получает информацию от внешнего мира, внешней среды с помощью чувствительных аппаратов, воспринимающих сигналы, то есть анализаторами.

Основная характеристика анализаторов – высокая чувствительность, хотя не каждый раздражитель, действующий на анализатор, вызывает ощущение. Чтобы ощущение проявилось, необходима определенная интенсивность раздражителя. Воздействие, превышающее предел интенсивности, вызывает боль и нарушение деятельности анализаторов.

В центральную нервную систему в процессе труда поступает большой объем информации. Мозг ее перерабатывает и сортирует по степени важности. Самыми важными являются сигналы, имеющие значение для жизнедеятельности всего организма; затем воспринимаются сигналы, имеющие значение для отдельных систем и органов.

Задача правильного проектирования технологического процесса – не допустить перенапряжения анализаторов, которое может привести к стрессам [10].

Одним из заболеваний нервной системы, связанных с перенапряжением

анализаторов, является координаторный невроз. Это своеобразное профессиональное заболевание наблюдается у операторов ПЭВМ, офисных работников, телеграфистов, музыкантов и др. Заболевание характеризуется нарушением функции двигательного анализатора по обеспечению координации строго дифференцированных движений рук. В результате повторяющихся движений в быстром темпе возникают условия для перенапряжения специфической функции руки, лежащей в основе профессиональной деятельности.

Монотонность – это психическое состояние человека, вызванное однообразием восприятия или действий. Общими признаками для всех видов монотонии являются перегрузка информацией при выполнении работы или, наоборот, ее недостаток, что накладывает на состояние человека определенный отпечаток: работник теряет интерес к выполняемой работе и у него возникает состояние, которое называется производственной скукой. Монотонная работа ухудшает экономические показатели, повышается травматизм, аварийность, растет текучесть кадров.

Основные меры по уменьшению влияния монотонности на человека:

- проектирование технического процесса, каждой трудовой операции, необходимых производству, так, чтобы сделать каждую операцию содержательной, вызывающей интерес у исполнителей;
- осуществлять перевод рабочих с одной на другую производственные операции в течение смены или рабочей недели;
- применять оптимальные режимы труда и отдыха в течение рабочего дня. При монотонной работе целесообразны частые, но короткие перерывы – от 2 до 5 минут через час или полчаса работы;
- не допускать принудительный темп работы: быстрый или медленный;
- необходимо соблюдать эстетичность производства, можно осуществлять музыкальное оформление производственного процесса.

Эмоциональные перегрузки характеризуются таким состоянием сотрудника, как утомление. Под утомлением принято понимать функциональное состояние, которое возникает на временной основе под воздействием определенных факторов и причин, формирующихся при продолжительной и интенсивной работе и приводящих к существенному снижению ее результативности. Основными проявлениями утомления можно назвать общее снижение выносливости и силы, ухудшение координации движений, существенное увеличение временных затрат на выполнение обычной работы и т. п.

Регулярное исполнение работы в условиях недостаточного восстановления, нерационально организованный труд и нервно-психическое напряжение на практике непременно ведут к переутомлению, вследствие чего возникает перенапряжение нервной системы, обострение различных заболеваний. Базисом всех указанных выше отклонений является нарушение возбuditельно-тормозных нервных процессов.

Важно понимать, что переутомление чрезмерно опасно для человека, оно появляется в результате продолжительного отсутствия полноценного отдыха, когда в рамках определенного периода времени не происходит восстановления организма. Переутомление накладывает отпечаток как на физическое, так и на эмоциональное состояние человека, что приводит к значительному снижению эффективности его деятельности [11].

Таким образом, основными последствиями эмоциональных перегрузок являются расстройство сна, повышенная возбудимость, рассеянность.

Ликвидация утомления возможна только при повышении общего и специального уровня тренированности организма, рационализации физической, интеллектуальной и эмоциональной активности. Профилактикой умственной усталости служит активизация таких сторон психической и физической активности, которые не связаны непосредственно с теми, которые привели к формированию данного состояния организма. Поэтому необходимы активный

отдых, переключения с одной работы на другую, применение средств восстановления. Оптимальное и разумное сочетание и чередование физических нагрузок и отдыха позволяет не только формировать и сохранять, но и развивать способность организма к наиболее эффективному и менее затратному процессу восстановления.

Для восстановления организма на практике часто применяются такие средства, как массаж, рациональное питание, витамины и т. д. Помимо этого, очень важно уделять повышенное внимание особенностям организации занятий физическими нагрузками [12].

Ритмопластические упражнения применяются, в том числе, в неврологической практике, выполняются в музыкальном сопровождении с заданными ритмом и тональностью, в зависимости от функционального состояния организма, типа высшей нервной деятельности, возраста и толерантности к нагрузке.

Спортивно-прикладные упражнения (это ходьба, бег, лазанье и ползание, плавание и многое другое) повышают тонус центральной нервной системы. Наиболее широко в практике лечебной физической культуры используется ходьба — упражнение, восстанавливающее опороспособность, улучшающее подвижность суставов и укрепляющее мышцы средних конечностей, стимулирующее вегетативные функции, восстанавливающее адаптацию к нагрузкам различной интенсивности. Ходьба может выполняться с разгрузкой и в обычном режиме, может применяться как специальное упражнение или в форме прогулок.

Для достижения оптимального эффекта от занятий физическими упражнениями необходимо учитывать следующие факторы:

- индивидуальные особенности занимающихся (возраст, пол, состояние здоровья, физическое развитие, уровень физической подготовленности);
- особенности физических упражнений (сложность, новизна,

специализированность, техническая характеристика);

— внешние условия: режим труда, конкретные условия двигательной деятельности (метеорологические условия, качество оборудования и инвентаря, гигиена мест занятий) [13].

Стоит отметить, что повторные нагрузки могут принести максимальный эффект исключительно на стадии повышенной работоспособности, а слишком длительные периоды отдыха значительно снижают результативность тренировок.

Таким образом, можно сформулировать основные правила борьбы с перегрузками:

— рационально организовать режим дня со сменой деятельности и активным отдыхом;

— уменьшить количество нагрузок на организм, необходимо объективно оценивать собственные силы и рационально их направлять;

— грамотно распоряжаться потоком информации;

— рационализировать питание;

— увеличить число пеших прогулок. Если 3÷5 раз в неделю совершать пешие прогулки по 20÷30 минут не позднее чем за 2 часа до сна, что уже через 1÷2 недели восстанавливается нарушенный сон и увеличивается работоспособность;

— больше времени проводить на солнце. Солнечные лучи существенно увеличивают выработку в организме серотонина, который положительно воздействует на организм человека в целом;

— полноценно отдыхать. Сон должен быть не менее 7÷8 часов в день [14].

Заключение

В начале дипломного проектирования был проведен анализ возможных угроз веб-сайта, определен перечень наиболее актуальных угроз.

Далее были подобраны необходимые меры защиты против проанализированных угроз и предложены рекомендации по защите.

Затем были разработаны программное обеспечение и конфигурация веб-сайта, которые используют предложенные рекомендации по защите.

На завершающем этапе дипломного проектирования было проведено функциональное тестирование подсистемы информационной безопасности разработанного веб-сайта. Тестирование показало, что для разработанного веб-сайта ни одна из рассмотренных угроз не была успешно реализована.

Таким образом, задание на дипломное проектирование было выполнено в полном объеме.

Список используемых источников

- 1 OWASP: Top 10 Web Application Security Risks [Электронный ресурс]. URL: <https://owasp.org/www-project-top-ten/> - Режим доступа: свободный. - (Дата обращения: 14.02.2021)
- 2 Laravel: The PHP Framework for Web Artisans [Электронный ресурс]. URL: <https://laravel.com/> - Режим доступа: свободный. - (Дата обращения: 21.03.2021)
- 3 Laravel Sanctum [Электронный ресурс]. URL: <https://laravel.com/docs/8.x/sanctum> - Режим доступа: свободный. - (Дата обращения: 21.03.2021)
- 4 Vue [Электронный ресурс]. URL: <https://ru.vuejs.org/index.html> - Режим доступа: свободный. - (Дата обращения: 21.03.2021)
- 5 Vuex [Электронный ресурс]. URL: <https://vuex.vuejs.org/ru/guide/> - Режим доступа: свободный. - (Дата обращения: 09.04.2021)
- 6 Let's Encrypt [Электронный ресурс]. URL: <https://letsencrypt.org/ru/> - Режим доступа: свободный. - (Дата обращения: 15.05.2021)
- 7 ГОСТ 12.0.003-2015. Система стандартов безопасности труда (ССБТ). Опасные и вредные производственные факторы. Классификация. - Приказом Федерального агентства по техническому регулированию и метрологии от 9 июня 2016 г. N 602-ст введен в действие в качестве национального стандарта Российской Федерации с 1 марта 2017 г. - Взамен ГОСТ 12.0.003-74. - М.: Стандартинформ, 2016.
- 8 Перегрузки эмоциональные и умственные. - [Электронный ресурс]. - URL: <https://studfiles.net/preview/5638826/page:11/>. - Режим доступа: свободный. - (Дата обращения: 22.05.2021).
- 9 Умственное переутомление. - [Электронный ресурс]. - URL: <http://nataliaprahova.ru/soveti-psiologa/umstvennoe-pereutomlenie/>. - Режим доступа: свободный. - (Дата обращения: 22.05.2021).

10 Перенапряжение анализаторов. - [Электронный ресурс]. - URL: <https://studfiles.net/preview/5638826/page:12/>. - Режим доступа: свободный. - (Дата обращения: 22.05.2021).

11 Физические перегрузки и функциональное перенапряжение организма работников. - [Электронный ресурс]. - URL: <http://www.kiout.ru/info/publish/27857> - Режим доступа: свободный. - (Дата обращения: 23.05.2021).

12 Шохин В.Е. Физические и эмоциональные перегрузки в процессе освоения программ профессиональной подготовки. Вестник краснодарского университета МВД России, 2017, № 1 (35), 183-185.

13 Лечебная физическая культура [Текст] : учеб.пособие / Виталий Александрович Епифанов. - М. : ГЭОТАР-МЕД, 2002. - 560 с.

14 Шохин В.Е. Физические и эмоциональные перегрузки в процессе освоения программ профессиональной подготовки. Вестник краснодарского университета МВД России, 2017, № 1 (35), 185-186.