Федеральное агентство связи

федеральное государственное бюджетное образовательное учреждение

высшего образования

«Сибирский государственный университет телекоммуникаций

и информатики»

(СибГУТИ)

Кафедра БиУТ

Допустить к защите зав. кафедрой

_____ /С.Н. Новиков /

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА СПЕЦИАЛИСТА

Разработка единой защищенной системы идентификации, аутентификации и авторизации для web-приложений

Пояснительная записка

Студент _____ / А.А. Зайцев /

Факультет АЭС Группа АБ-56

Руководитель _____ / А.А. Буров /

Консультанты:

– по экономическому обоснованию

_____ / И.С. Мухина /

– по безопасности жизнедеятельности

_____ / Н.Н. Симакова /

Рецензент: _____ / И.В. Нечта /

Новосибирск 2021

Под. и дата

Инв. № дубл.

Взам. инв. №

Подп. и дата

Инв. № подп.

Федеральное агентство связи

федеральное государственное бюджетное образовательное учреждение

высшего образования

«Сибирский государственный университет телекоммуникаций и информатики»

(СибГУТИ)

КАФЕДРА

Безопасность и управление в телекоммуникациях

ЗАДАНИЕ

НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ СПЕЦИАЛИСТА

СТУДЕНТА А.А Зайцева ГРУППЫ АБ-56

«УТВЕРЖДАЮ»

« 28 » июля 2020 г.

Зав. кафедрой БиУТ

_____/ С.Н. Новиков /

Новосибирск 2020

1. Тема выпускной квалификационной работы специалиста:

Разработка единой защищенной системы идентификации, аутентификации и авторизации для web-приложений

утверждена приказом по университету от « 28 » июля 2020 г. № 4/1011о-20

2. Срок сдачи студентом законченной работы « 15 » января 2020 г.

3. Исходные данные по проекту (эксплуатационно-технические данные, техническое задание):

Язык программирования Java и его документация

Framework Spring и его документация

Документация протокола OAuth2

Документация протокола OpenID Connect

4. Содержание расчетно-пояснительной записки (перечень подлежащих разработке вопросов)	Сроки выполнения по разделам
Введение	13.09.2020 г.
1. Проектирование единой защищенной системы идентификации, аутентификации и авторизации для web-приложений	11.10.2020 г.
2. Анализ протокола OAuth2	08.11.2020 г.
3. Анализ атак на авторизационный сервер	06.12.2020 г.
4. Разработка единой защищенной системы идентификации, аутентификации и авторизации для web-приложений	13.12.2020 г.
5. Безопасность жизнедеятельности	17.12.2020
6. Технико-экономическое обоснование работы	20.12.2020 г.
7. Заключение	27.12.2020 г.
8. Список литературы	09.01.2020 г.
9. Приложения	15.01.2021 г.

Консультанты по ВКР (с указанием относящихся к ним разделов):

1. Раздел по технико-экономическому обоснованию

_____

_____

_____

_____/_____/

2. Раздел по безопасности жизнедеятельности

_____

_____

_____

_____

_____/ _____/

Дата выдачи задания	Задание принял к исполнению
« 01 » сентября 2020 г.	« 01 » сентября 2020 г.
_____/ А.А. Буров/	_____/ А.А. Зайцев/
(подпись, Ф.И.О. руководителя)	(подпись, Ф.И.О. студента)

Федеральное агентство связи

федеральное государственное бюджетное образовательное учреждение

высшего образования

«Сибирский государственный университет телекоммуникаций и информатики»

(СибГУТИ)

РЕЦЕНЗИЯ

на выпускную квалификационную работу студента А.А. Зайцева

по теме «Разработка единой защищенной системы идентификации, аутентификации и авторизации для web-приложений»

Представленная к рецензии работа Зайцева А.А. посвящена идентификации, аутентификации и авторизации в web-приложениях. Выбранная тема ВКР является актуальной. В работе описываются различные атаки на web-системы. Выполнен анализ стандартов аутентификации и протокола OAuth2. Предложенная защищенная система является устойчивой к различным уязвимостям.

К тексту пояснительной записки имеются следующие замечания:

1. в разделе 4.4 Разработка архитектуры приложения не приведена сама архитектура, а только алгоритмы работы системы;

2. в тексте имеются слишком короткие подразделы, например 1.1, состоящие из одного предложения;

3. в каждой главе делается постановка задачи по данной главе, тогда как следовало бы сделать общую постановку в начале работы и последовательно ее выполнять.

Считаю, что работу заслуживает оценки **хорошо**, а ее автор, Зайцев А.А., присуждения квалификации специалист.

Доц. каф. ПМиК, к.т.н. Нечта Иван Васильевич

« 18 » января 2021 г.

С Рецензией ознакомлен _____/А.А. Зайцев/

« 18 » января 2021 г.

Федеральное агентство связи

федеральное государственное бюджетное образовательное учреждение

высшего образования

«Сибирский государственный университет телекоммуникаций и информатики»

(СибГУТИ)

ОТЗЫВ

о работе студента А.А. Зайцева в период подготовки выпускной квалификационной работы по теме «Разработка единой защищенной системы идентификации, аутентификации и авторизации для web-приложений»

Работа имеет практическую ценность		Тема предложена предприятием	
Работа внедрена		Тема предложена студентом	
Рекомендую работу к внедрению		Тема является фундаментальной	
Рекомендую работу к опубликованию		Рекомендую студента в магистратуру	
Работа выполнена с применением ЭВМ		Рекомендую студента в аспирантуру	

Руководитель выпускной квалификационной работы специалиста

Доц. каф. БиУТ, к.т.н. Буров Артем Анатольевич

« 15 » января 2021 г.

С Отзывом ознакомлен /А.А. Зайцев/

« 15 » января 2021 г.

Приложение к Отзыву

Уровень сформированности компетенций у студента

А.А. Зайцева

Компетенции		Уровень сформированности компетенций		
		высокий	средний	низкий
1		2	3	4
Профессиональные	ПК-1 - способностью осуществлять анализ научно-технической информации, нормативных и методических материалов по методам обеспечения информационной безопасности телекоммуникационных систем			
	ПК-5 - способностью проектировать защищенные телекоммуникационные системы и их элементы, проводить анализ проектных решений по обеспечению заданного уровня безопасности и требуемого качества обслуживания, разрабатывать необходимую техническую документацию с учетом действующих нормативных и методических документов			
	ПК-7 - способностью осуществлять рациональный выбор средств обеспечения информационной безопасности телекоммуникационных систем с учетом предъявляемых к ним требований качества обслуживания и качества функционирования			
	ПК-12 - способностью выполнять технико-экономические обоснования, оценивать затраты и результаты деятельности организации в области обеспечения информационной безопасности			

АННОТАЦИЯ

Выпускной квалификационной работа студента А.А. Зайцева
по теме Разработка единой защищенной системы идентификации, аутентификации и авторизации для web-приложений

Объём работы – 113 страниц, на которых размещены 17 рисунков и 4 таблиц. При написании работы использовалось 25 источников.

Ключевые слова: CSRF, XSS, DoS, атака, brute force, идентификация, аутентификация, авторизация, OAuth2, OpenID Connect.

Работа выполнена на: кафедре БиУТ СибГУТИ

Руководитель: доц. каф. БиУТ Буров А.А.

Целью работы разработка единой защищенной системы идентификации, аутентификации и авторизации для web-приложений

Решаемые задачи: проектирование единой защищенной системы идентификации, аутентификации и авторизации для web-приложений, анализ протокола OAuth2, анализ атак на авторизационный сервер, разработка защищенной системы идентификации, аутентификации и авторизации, безопасность жизнедеятельности, технико-экономическое обоснование работ

Основные результаты: разработанная единая защищенная система идентификации, аутентификации и авторизации для web-приложений

Graduation thesis abstract

of A.A.Zaytsev on the theme Develop unified secure system of identification, authentication and authorization for web applications

The paper consists of 113 pages, with 17 figures and 4 tables/charts/diagrams. While writing the thesis 25 reference sources were used.

Keywords: CSRF, XSS, DoS, attack, brute force, identification, authentication, authorization, OAuth2, OpenID Connect

The thesis was written at BIUT department SibSUTIS
(name of organization or department)
Scientific supervisor associate professor of the BiUT Burov Artyom

The goal/subject of the paper is development of a unified secure system of identification, authentication and authorization for web applications

Tasks: design of a unified secure identification, authentication and authorization system for web applications, analysis of the OAuth2 protocol, analysis of attacks on an authorization server, development of a secure identification, authentication and authorization system, life safety, technical and economic justification of work

Results developed unified secure system of identification, authentication and authorization for web applications

ОГЛАВЛЕНИЕ

Введение

В современном мире почти все приложения, которые работают в сети Интернет реализуют какие-либо системы идентификации, аутентификации и авторизации. Web-ресурсы, которые заботятся об удобстве пользователей, постоянно упрощают данные системы для клиента, оставляя или даже улучшая степень своей защиты. Одной из наиболее безопасной и удобной системой авторизации является система, которая реализует протокол OAuth2.

Целью данного проекта является разработка единой системы идентификации, аутентификации и авторизации, легко внедряемой в уже существующие web-приложения.

Для этого необходимо:

- спроектировать единую защищенную систему идетификации, аутентификации и авторизации для web-приложений;
- провести анализ протокола OAuth2;
- провести анализ атак на авторизационный сервер;
- разработать защищенную систему идентификации, аутентификации и авторизации;
- рассмотреть вопросы безопасности жизнедеятельности;
- выполнить технико-экономические расчеты.

1 Проектирование единой защищенной системы идентификации, аутентификации и авторизации для web-приложений

1.1 Постановка задачи

В данной главе необходимо провести анализ стандартов аутентификации для построения SSO и описать основную функциональность единой защищенной системы идентификации, аутентификации и авторизации.

1.2 Анализ стандартов аутентификации для построения SSO

Система единого входа (SSO, Sigle Sign-On) — это отдельный тип продуктов или встраиваемая технология, позволяющая не прибегать к повторной аутентификации пользователя при его переходе по различным разделам и сервисам одного портала, таким как форум, блог и другие, или при работе с несколькими приложениями. Другими словами, пользователь проходит процедуру аутентификации в одном месте, после чего получает доступ ко всем связанным разделам, и ему не приходится вводить свои учетные данные в нескольких формах.[18]

Стандарт Security Assertion Markup Language (SAML) описывает способы взаимодействия и протоколы между identity provider и service provider для обмена данными аутентификации и авторизации посредством токенов. Изначально версии 1.0 и 1.1 были выпущены в 2002 – 2003 гг., в то время как версия 2.0, значительно расширяющая стандарт и обратно несовместимая, опубликована в 2005 г. Этот основополагающий стандарт достаточно сложный и поддерживает много различных сценариев интеграции систем. Основные «строительные блоки» стандарта:

— assertions – собственный формат SAML токенов в XML формате;

— protocols – набор поддерживаемых сообщений между участниками;

— bindings – механизмы передачи сообщений через различные транспортные протоколы;

— profiles – типичные сценарии использования стандарта.

WS-Trust и WS-Federation входят в группу стандартов WS-*, описывающих SOAP/XML-веб сервисы. Они разрабатываются группой компаний, куда входят Microsoft, IBM, VeriSign и другие. Наряду с SAML, эти стандарты достаточно сложные, используются преимущественно в корпоративных сценариях.

Стандарт WS-Trust описывает интерфейс сервиса авторизации, именуемого Secure Token Service (STS). Он работает по протоколу SOAP и поддерживает создание, обновление и аннулирование токенов. При этом стандарт допускает использование токенов различного формата, однако на практике в основном используются SAML-токены.

Стандарт WS-Federation касается механизмов взаимодействия сервисов между компаниями, в частности, протоколов обмена токенами. При этом WS-Federation расширяет функции и интерфейс сервиса STS, описанного в стандарте WS-Trust. Среди прочего, стандарт WS-Federation определяет:

— формат и способы обмена метаданными о сервисах;

— функцию единого выхода из всех систем (single sign-out);

— сервис атрибутов, предоставляющий дополнительную информацию о пользователе;

— сервис псевдонимов, позволяющий создавать альтернативные имена пользователей;

— поддержку пассивных клиентов (браузеров) посредством перенаправления пользователя.

Можно сказать, что WS-Federation позволяет решить те же задачи, что и SAML, но их подходы и реализация отличаются.

В отличие от SAML и WS-Federation, стандарт OAuth (Open Authorization) не описывает протокол аутентификации пользователя. Вместо этого он определяет механизм получения доступа одного приложения к другому от имени пользователя. Для осуществления аутентификации пользователя на базе этого стандарта необходимо:

— реализовать метод на сервере авторизации, предоставляющий информацию о самом пользователе;

— использовать стандарт OpenID Connect, разработанный как слой учетных данных поверх OAuth [1].

Первая версия стандарта разрабатывалась в 2007 – 2010 гг., а текущая версия 2.0 опубликована в 2012 г. OAuth 2.0 — это полная переработка OAuth 1.0 с нуля, разделяющая только общие цели и общий пользовательский опыт. OAuth 2.0 не имеет обратной совместимости с OAuth 1.0 или 1.1, его следует рассматривать как совершенно новый протокол [24].

1.3 Описание функциональности ЕЗСИАА для web-приложений

Единый защищенный модуль идентификации, аутентификации и авторизации (ЕЗСИАА) является реализацией протокола OAuth2 и его надстройки OpenID Connect.

Основная идея сервиса заключается в том, что подключенные к нему web-приложения, с согласия пользователя ЕЗСИАА, могут получить доступ к защищенным данным. При этом web-ресурс не получает логин и пароль человека.

Основные возможности ЕЗСИАА:

— регистрационные действия;

— авторизационные и аутентификационные действия;

— изменение учетных данных;

— получение защищенной информации пользователя.

Регистрационные действия подразумевают под собой регистрацию стороннего web-сервиса, который сможет получать защищённые данные пользователя, а также регистрацию самого пользователя в системе ЕЗСИАА.

Авторизационные и аутентификационные действия — это действия, направленные на проверку введённых данных и выдачу прав клиенту или пользователю.

Изменение учетных данных – функциональность, которая позволяет пользователю устанавливать или изменять данные о нем в ЕЗСИАА.

1.4 Выводы по разделу

В данной главе был проведен анализ стандартов аутентификации для построения SSO, а также была описана основная функциональность ЕЗСИАА.

2 Анализ протокола OAuth2

2.1 Постановка задачи

В этой главе необходимо провести обзор архитектуры протокола OAuth2, проанализировать его уязвимости, а также методы защиты от них.

2.2 Обзор архитектуры протокола OAuth2

OAuth – открытый протокол для делегирования доступа, который позволяет предоставить третьей стороне ограниченный доступ к защищённым ресурсам пользователя без необходимости передавать ей логин и пароль [2].

Протокол OAuth2 имеет 4 основных роли:

— владелец ресурсов;

— сервер ресурсов;

— клиент;

— сервер авторизации.

Владелец ресурсов является пользователем, который имеет данные на сервере ресурсов.

Сервер ресурсов – сервер, на котором размещены защищенные ресурсы пользователя.

Сервер авторизации – сервер, выдающий токены доступа после успешной аутентификации и авторизации владельца ресурса.

Сервер ресурсов и сервер авторизации может быть единой сущностью.

Клиент – приложение, которое с помощью токена доступа (access token) запрашивает защищенные ресурсы от имени владельца ресурса [3].

Прежде чем приложение сможет получать защищенные ресурсы, его необходимо зарегистрировать на авторизационном сервере. Во время регистрации авторизационный сервер создаст учетную запись приложения, сгенерировав идентификатор приложения, а также приватный токен приложения.

Существует несколько потоков (сценариев взаимодействия сторон):

— поток с кодом авторизации;

— поток неявного доступа;

— поток с предоставлением клиенту пароля;

— поток клиентских полномочий.

Сценарий с кодом авторизации является самым распространенным сценарием. Он используется приложениями, у которых есть backend (программный слой, отвечающий за обработку данных).

Этапы получения токена доступа в сценарии с кодом авторизации:

1. пользователю предоставляется ссылка на авторизационный сервер с параметрами запроса:

— client_id (идентификатор приложения);

— redirect_uri (ссылка для редиректа на страницу после успешной авторизации);

— response_type (схема взаимодействия сторон) со значением code;

— scope (уровень доступа к защищенным ресурсам, если на сервере авторизации их несколько);

2. пользователь авторизуется на сервере авторизации и дает разрешение клиенту на доступ к своей защищенной информации;

3. авторизационный сервер генерирует код авторизации и совершает перенаправление пользователя на страницу, которую клиент передал в значении параметра «redirect_uri» вместе с параметром запроса «code», где значением является сгенерированный код авторизации.

4. приложение делает POST запрос в сервер авторизации, где передает идентификатор клиента, приватный токен клиента, код авторизации для аутентификации клиента и генерации токена доступа;

5. авторизационный сервер возвращает токен доступа клиенту.

Схематичный процесс авторизации с помощью потока с кодом авторизации представлен на рисунке 2.1.

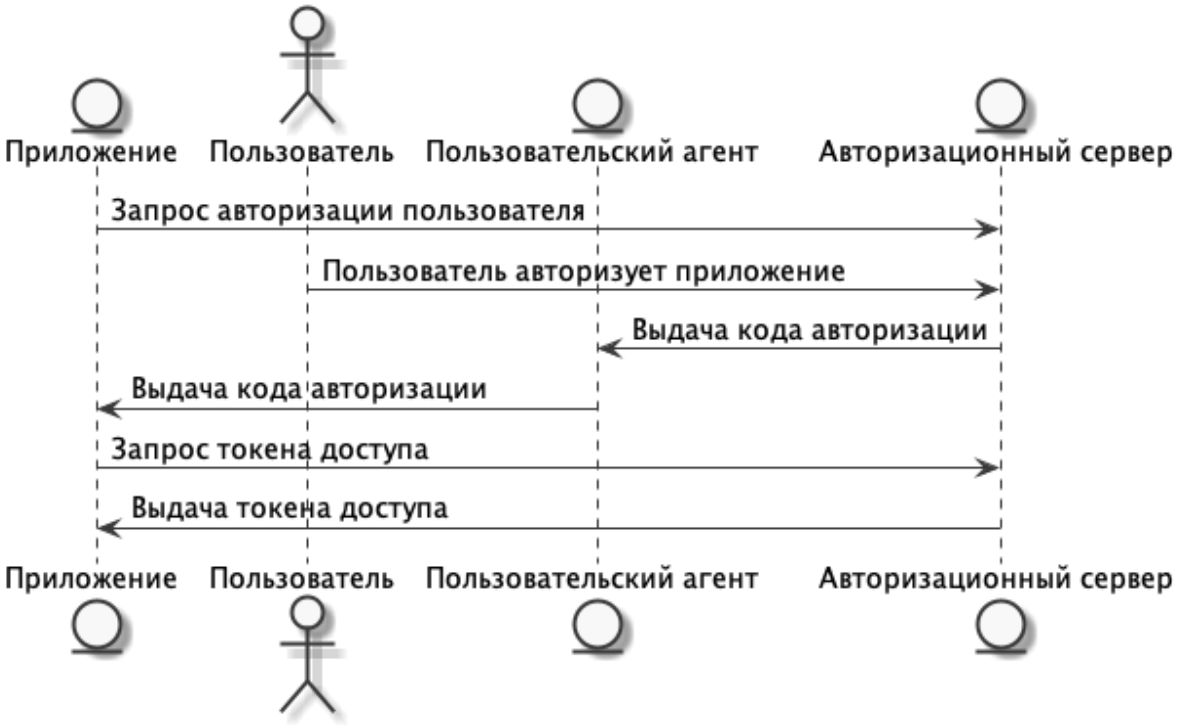

Рисунок 2.1 – Процесс авторизации с помощью потока с кодом подтверждения

Сценарий взаимодействия с неявным потоком применяется для мобильных и SPA (single page application) приложений, где конфиденциальность приватного токена клиента не может быть гарантирована.

Этапы получения токена доступа в сценарии с неявным доступом:

1. пользователю предоставляется ссылка на авторизационный сервер с параметрами запроса:

— client_id;

— redirect_uri;

— response_type со значением token;

— scope;

2. пользователь авторизуется на сервере авторизации и дает разрешение клиенту на доступ к своей защищенной информации;

3. авторизационный сервер перенаправляет пользовательский агент (браузер) по ссылке, указанной в параметре запроса «redirect_uri», сохраняя при этом token в параметрах запроса;

4. браузер следует по URI перенаправления, сохраняя при этом токен доступа;

5. клиент возвращает веб-страницу, которая содержит скрипт для извлечения токена доступа;

6. пользовательский агент запускает скрипт извлечения токена доступа, а затем передаёт извлечённый токен клиенту.

Схематичный процесс авторизации с помощью неявного потока представлен на рисунке 2.2.

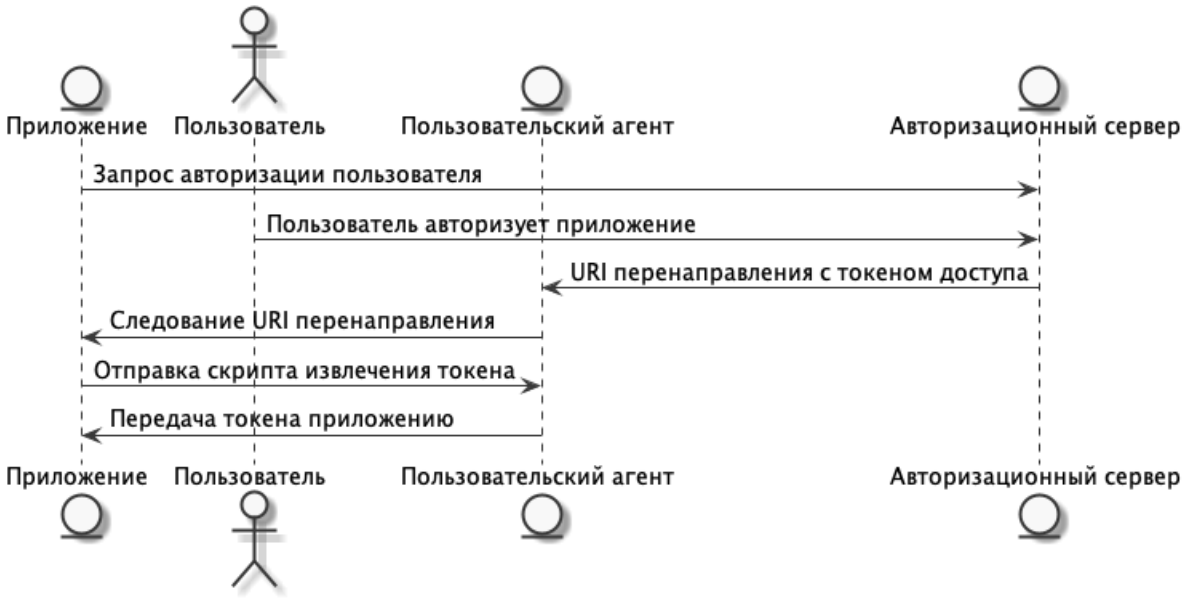

Рисунок 2.2 – Процесс авторизации с помощью неявного потока

Если используется сценарий взаимодействия с предоставлением клиенту пароля, то пользователь передает свой логин и пароль приложению, который затем отправляется авторизационному серверу. В свою очередь, он сгенерирует и вернет токен доступа. Данная схема авторизации может использоваться только в том случае, если клиент является частью самого сервиса.

Сценарий взаимодействия клиентских полномочий позволяет приложению осуществлять доступ к своему собственному аккаунту сервиса. Это может быть полезно, например, когда клиент хочет обновить собственную регистрационную информацию на сервисе или URI перенаправления [3]. В этом сценарии клиент передает авторизационному серверу идентификатор и приватный токен.

2.3 Анализ уязвимостей протокола OAuth2 и методов защиты

Уязвимым местом данного протокола является возможность перехватить код авторизации. При получении ответа на запрос кода активации, он может быть перехвачен, например, вредоносным расширением браузера, и перенаправлен злоумышленнику, который может обменять полученный код авторизации на токен для доступа к защищенным ресурсам пользователя. Схема атаки представлена на рисунке 2.3.

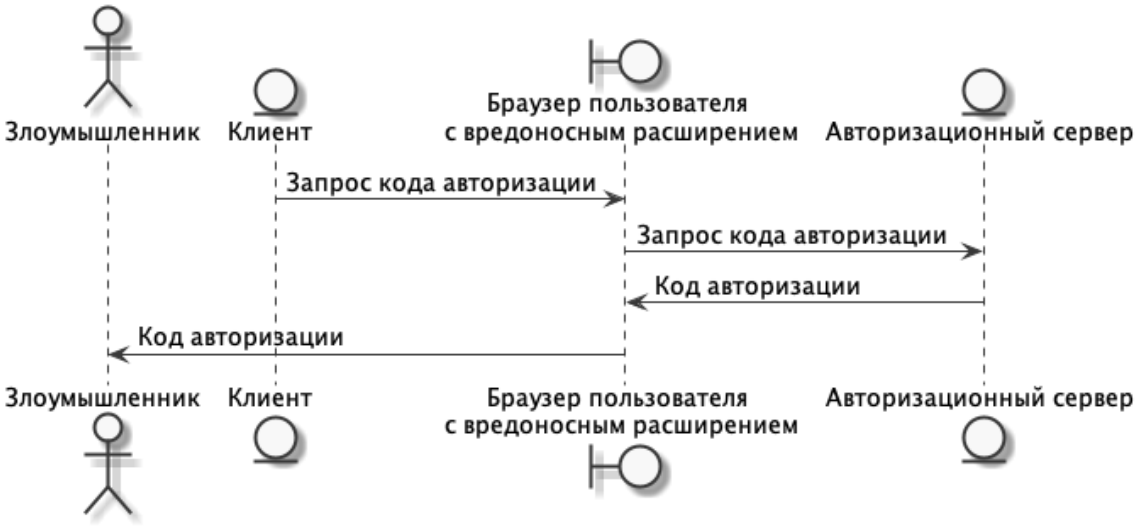

Рисунок 2.3 – Схема атаки на авторизационный сервер для получения кода авторизации

Способом защиты от данного типа атак является следующая схема организации запроса access token. Перед тем, как запрашивать авторизационный код, клиент генерирует и сохраняет на своей стороне проверочный код (code_verifier). С помощью заданного алгоритма (название этого алгоритма соответствует коду

code_challenge_method) хэширует проверочный код, получая тем самым захэшированный проверочный код (code_challenge). Клиент добавляет к запросу авторизационного кода захэшированный проверочный код и код алгоритма хэширования.

Когда клиент запрашивает токен доступа у авторизационного сервера, он добавляет проверочный код. В свою очередь, авторизационный сервер хэширует присланный проверочный код и сравнивает с переданным ранее захэшированным проверочным кодом. Если они идентичны, то сервер считает клиента аутентифицированным и высылает токен доступа [25].

Атака redirect validation URI возможна, если сервер авторизации поддерживает wildcard в redirect URI при регистрации клиента. Даная функциональность позволяет злоумышленнику создать поддомен, подходящий под wildcard легального клиента и зарегистрировать своего на сервере авторизации. Таким образом, он сможет отправить пользователя напрямую по ссылке на сервер авторизации или посредством атаки open redirect, в которую встроен redirect URI вредоносного клиента и получить код авторизации. Для защиты от этой уязвимости необходимо запретить wildcard для поддоменов и строго сверять полученный redirect uri с адресом, который был указан при регистрации клиента.

2.4 Выводы по разделу

В данной главе был проведен анализ архитектуры протокола OAuth2, были проанализированы уязвимости данного протокола, а также способы их устранения.

3 Анализ атак на авторизационный сервер

3.1 Постановка задачи

В этой главе необходимо провести анализ атак, которым может быть подвергнут авторизационный сервер и способы защиты от них.

Авторизационный сервер является обычным web-приложением, поэтому он может быть подвергнут следующим атакам:

— CSRF атаки;

— Brute-Force атаки;

— DoS атаки;

— XSS атаки.

3.2 Анализ атаки CSRF

CSRF – опаснейшая атака, которая приводит к тому, что злоумышленник может выполнить на неподготовленном сайте массу различных действий от имени других, зарегистрированных посетителей [4].

Атака CSRF работает, потому что запросы браузера автоматически включают все файлы cookie. Следовательно, если пользователь аутентифицирован на сайте, сайт не может отличить законные запросы от поддельных.

Согласно аналитике Positive Technologies, CSRF атаки занимают 3 место из всех атак на web-приложения. Статистика представлена в таблице 3.1 [5].

Таблица 3.1 – Статистика атак на web-приложения

Название атаки	Процент от всех атак на web-приложения
SQL Injection	27
Path Traversal	17

Продолжение таблицы 3.1

Cross-Site Scripting	14
Local File Inclusion	11
Inforamation Leakage	8
OS Commanding	7
Brute Force	5
Remote Code Execution	2
Denial of Service	2
Server Side Template Injection	1
Другие	6

CSRF атаки используются злоумышленником, чтобы заставить целевую систему выполнять функцию через браузер жертвы без ведома жертвы, по крайней мере, до тех пор, пока не будет совершена несанкционированная транзакция.

Реализовать защиту от данного вида атак позволяет CSRF token, требования к которому:

— уникальный токен для каждой операции;

— действует единожды;

— имеет размер, устойчивый к подбору;

— сгенерирован криптографически стойким генератором псевдослучайных чисел;

— имеет ограниченное время жизни.

Основные методы использования токенов для защиты web-приложений от CSRF атак:

— synchronizer Tokens (statefull);

— double submit cookie (stateless);

— encrypted token (stateless).

Суть Synchronizer Tokens заключается в том, что в начале новой сессии на стороне сервера генерируется токен и кладется в хранилище данных сессии. При

последующих запросах клиент обязан передать токен серверу для проверки. Данный метод отличается от double submit cookie и encrypted token тем, что он требует хранить данные на стороне сервера.

Double submit cookie не требует хранить токен на сервере. При запросе от клиента на стороне сервера генерируется токен, при этом он добавляется и в cookie, и в один из заголовков. В последующих запросах клиент обязан предоставлять оба, полученных ранее, токена.

Encrypted Token достаточно прост, так как при запросе от клиента на стороне сервера генерируется токен, в котором зашифрована временная метка и идентификатор пользователя. В последующих запросах клиент обязан предоставлять полученный ранее токен, а сервер валидировать его, предварительно расшифровав [6].

3.3 Анализ Brute-force атаки

Brute force называется метод взлома учетных записей путем подбора паролей к ним. Данная атака возможна из-за того, что пользователи сети Интернет часто используют не надежные, часто встречаемые пароли. Термин образован от англоязычного словосочетания «brute force», означающего в переводе «грубая сила». Суть подхода заключается в последовательном автоматизированном переборе всех возможных комбинаций символов с целью рано или поздно найти правильную [7].

На сервере авторизации можно зарегистрироваться, введя логин и пароль, что делает возможным brute-force атаки.

В случае подбора злоумышленником правильной комбинации логина и пароль, он может полностью управлять информацией, деньгами и т.д. пользователя. Данный тип атаки занимает 5 % всех атак на web-приложения (таблица 3.1)

Основные программы для реализации таких атак [21]:

— BurpSuite;

— Hydra;

— Patator;

— Nmap.

Способ защиты от brute force атак:

— максимальное количество попыток входа для одного хоста;

— максимальное количество попыток входа для одного пользователя;

— использовать капчу;

— удалить подсказки о неправильном вводе логина или пароля.

3.4 Анализ DoS атаки

Распределенные сетевые атаки часто называются распределёнными атаками типа «отказ в обслуживании» (Distributed Denial of Service, DDoS). Этот тип атаки использует определенные ограничения пропускной способности, которые характерны для любых сетевых ресурсов, например, инфраструктуре, которая обеспечивает условия для работы сайта компании. DDoS-атака отправляет на атакуемый веб-ресурс большое количество запросов с целью превысить способность сайта обрабатывать их все и вызвать отказ в обслуживании [8].

Авторизационный сервер, как и любое web-приложение, подвержен DoS атакам.

Сетевые ресурсы, такие как веб-серверы, имеют ограничения по количеству запросов, которые они могут обслуживать одновременно. Помимо допустимой нагрузки на сервер существуют также ограничения пропускной способности канала, который соединяет сервер с Интернетом. Когда количество запросов превышает производительность любого компонента инфраструктуры, может произойти следующее:

— существенное замедление времени ответа на запросы;

— отказ в обслуживании всех запросов пользователей или части из них.

Для отправки очень большого количества запросов на ресурс жертвы злоумышленник часто создает сеть из зараженных «зомби-компьютеров». Поскольку преступник контролирует действия каждого зараженного компьютера в зомби-

сети, атака может быть слишком мощной для веб-ресурса жертвы. Схема DDoS атаки представлена на рисунке 3.1 [8].

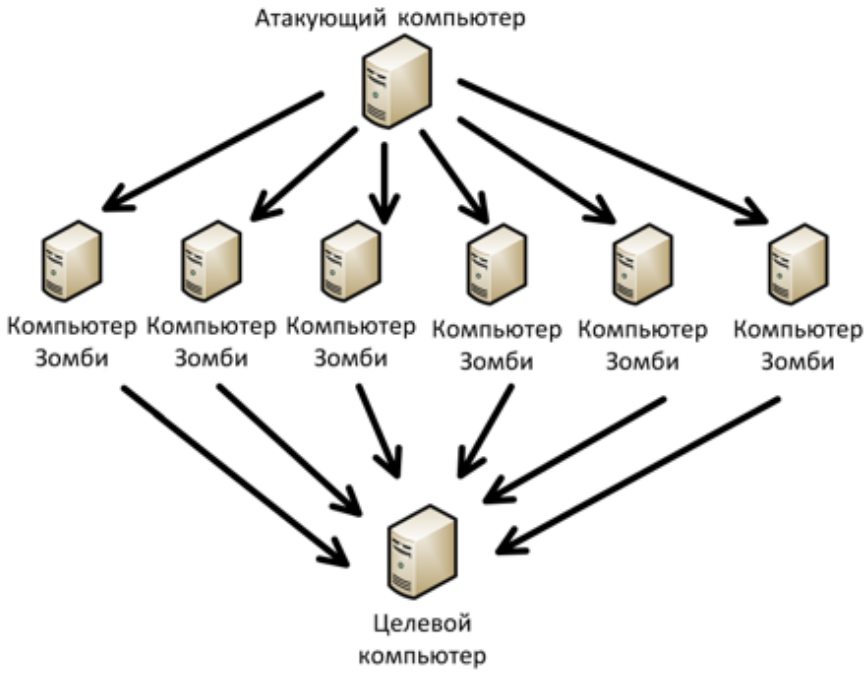

Рисунок 3.1 – Схема DDos атаки

DoS атаки занимают всего 2% всех атак на web-приложения (таблица 3.1), но при этом наносят огромный финансовый и репутационный урон из-за простоя сервиса.

В связи с тем, что практически каждый компьютер подключен к сети Интернет или к локальной сети, возможен такой тип атаки, как сетевой flood – атака, которая заключается в отправке большого количества бессмысленных или неправильно сформированных запросов к компьютерной системе или сетевому оборудованию с целью отказа оборудования из-за исчерпания системных ресурсов (процессора, памяти или каналов связи).

Атакующие прибегают к данному виду атаки для захвата таких ресурсов как оперативная и физическая память, процессорное время и т.д.

Недостаточная проверка данных пользователя может приводить к бесконечному или длительному циклу, что, как следствие, приводит к повышенному и

продолжительному потреблению процессорных ресурсов либо выделению больших объемов памяти, вплоть до ее исчерпания.

Атаки второго рода – это атаки, которые приводят к ложным срабатываниям систем защиты, тем самым приводят к недоступности определенные ресурсы.

Если злоумышленник отсылает небольшие http-пакеты, которые заставляют в свою очередь отвечать сервер пакетами, размеры которых значительно больше, то такая атака называется HTTP-flood. Тем самым злоумышленник имеет большой шанс насытить полосу пропускания жертвы и вызвать отказ в работе сервисов. Для того, чтобы ответные пакеты не вызывали отказ в обслуживании у атакующего, он подменяет свой сетевой адрес на адреса узлов в сети.

ICMP-flood (Smurf-атака) является одним из самых опасных. В ней по широковещательному адресу злоумышленник отправляет поддельный ICMP-пакет, в котором адрес атакующего меняется на адрес жертвы. Все узлы присылают ответ на данный ping-запрос. Для такого вида атаки обычно используют большую сеть, чтобы у компьютера-жертвы не было никаких шансов. Таким образом, запрос, отправленный через сеть в 1000 компьютеров, будет усилен в 1000 раз.

UDP flood (Fraggle) является аналогом ICMP flood, но вместо ICMP пакетов используются UDP пакеты. На седьмой порт жертвы отправляются echo-команды по широковещательному запросу. После чего подменяется ip-адрес злоумышленника на ip-адрес жертвы, которая получает множество ответных сообщений, что приводит к насыщению полосы пропускания и отказу в обслуживании жертвы.

SYN-flood атака основана на попытке запуска большого числа одновременных TCP-соединений через посылку SYN-пакета с несуществующим обратным адресом. После нескольких попыток отослать в ответ ACK-пакет на недоступный адрес большинство операционных систем ставят неустановленное соединение в очередь. И только после n-ой попытки закрывают соединение. Поскольку поток ACK-пакетов очень большой, вскоре очередь оказывается заполненной, и ядро дает отказ на попытки открыть новое соединение.

Атакующий отсылает пакеты серверу, которые не насыщают полосу пропускания, а тратят все его процессорное время. Соответственно, в системе может

пройти сбой, и легальные пользователи не смогут получить доступ к необходимым ресурсам.

При неправильной системе ротации лог-файлов и неправильно установленной системе квотирования злоумышленник может отправлять большие по объему пакеты, которые вскоре займут все свободное место на жестком диске сервера [9].

Существуют различные способы защиты от DDoS атак.

Одним из первых методов нейтрализации DDoS-атак является сведение к минимуму размера зоны, которую можно атаковать. Подобный прием ограничивает возможности злоумышленников для атаки и обеспечивает возможность создания централизованной защиты. Необходимо убедиться, что доступ к приложению или ресурсам не был открыт для портов, протоколов или приложений, взаимодействие с которыми не предусмотрено. Таким образом, сведение к минимуму количества возможных точек для атаки позволяет сосредоточить усилия на их нейтрализации. В некоторых случаях этого можно добиться, разместив свои вычислительные ресурсы за сетями распространения контента (CDN) или балансировщиками нагрузки и ограничив прямой интернет-трафик к определенным частям своей инфраструктуры, таким как серверы баз данных. Также можно использовать брандмауэры или списки контроля доступа (ACL), чтобы контролировать, какой трафик поступает в приложения.

Двумя основными элементами нейтрализации крупномасштабных DDoS-атак являются пропускная способность и производительность сервера, достаточная для поглощения и нейтрализации атак.

При проектировании приложений необходимо убедиться, что поставщик услуг хостинга предоставляет избыточную пропускную способность подключения к Интернету, которая позволяет обрабатывать большие объемы трафика. Поскольку конечная цель DDoS-атак – повлиять на доступность ресурсов или приложений, необходимо размещать их рядом не только с конечными пользователями, но и с крупными узлами межсетевого обмена трафиком, которые легко обеспечат вашим пользователям доступ к приложению даже при большом объеме трафика. Работа с интернет-приложениями обеспечивает еще более широкие воз-

можности. В этом случае можно воспользоваться сетями распространения контента (CDN) и сервисами интеллектуального преобразования адресов DNS, которые создают дополнительный уровень сетевой инфраструктуры для обслуживания контента и разрешения DNS-запросов из мест, которые зачастую расположены ближе к конечным пользователям.

Большинство DDoS-атак являются объемными и потребляют много ресурсов, поэтому важно иметь возможность быстро увеличивать или уменьшать объем своих вычислительных ресурсов. Это можно обеспечить, используя избыточный объем вычислительных ресурсов или ресурсы со специальными возможностями, такими как более производительные сетевые интерфейсы или улучшенная сетевая конфигурация, что позволяет поддерживать обработку больших объемов трафика. Кроме того, для постоянного контроля и распределения нагрузок между ресурсами и предотвращения перегрузки какого-либо одного ресурса часто используются соответствующие балансировщики. Сведения о типичном и нетипичном трафике. Каждый раз, когда обнаруживается повышение объема трафика, попадающего на хост, в качестве ориентира можно брать максимально возможный объем трафика, который хост может обработать без ухудшения его доступности. Такая концепция называется ограничением скорости. Более продвинутые методы защиты соответственно обладают дополнительными возможностями и могут интеллектуально принимать только трафик, который разрешен, анализируя отдельные пакеты. Для использования подобных средств необходимо определить характеристики хорошего трафика, который обычно получает целевой объект, и иметь возможность сравнивать каждый пакет с этим эталоном.

Развертывание брандмауэров для отражения сложных атак уровня приложений. Против атак, которые пытаются использовать уязвимость в приложении, например против попыток внедрения SQL-кода или подделки межсайтовых запросов, рекомендуется использовать Web Application Firewall (WAF) [22]. Кроме того, из-за уникальности этих атак, сервис должен быть способен самостоятельно нейтрализовать запрещенные запросы, которые могут иметь определенные характеристики, например, могут определяться как отличные от хорошего трафика или

исходить из подозрительных IP-адресов, из неожиданных географических регионов и т. д. Чтобы нейтрализовать происходящие атаки, иногда может быть полезно получить поддержку специалистов для изучения характеристик трафика и создания индивидуальной защиты [10].

3.5 Анализ XSS

Межсайтовый скриптинг (XSS) — широко распространенная уязвимость, затрагивающая множество web-приложений. Она позволяет злоумышленнику внедрить вредоносный код в web-сайт таким образом, что браузер пользователя, зашедшего на сайт, выполнит этот код [11].

Авторизационный сервер имеет несколько HTML форм, с помощью которых можно осуществить XSS атаку.

Существуют 3 типа XSS уязвимостей:

— хранимые (stored);

— отраженные (reflected);

— DOM-based.

Уязвимости, вызванные кодом на стороне сервера (Java, PHP, .NET и т. д.).

Отраженная XSS-атака срабатывает, когда пользователь переходит по специально подготовленной ссылке. Эти уязвимости появляются, когда данные, предоставленные веб-клиентом, чаще всего в параметрах HTTP-запроса или в форме HTML, исполняются непосредственно серверными скриптами для синтаксического анализа и отображения страницы результатов для этого клиента, без надлежащей обработки.

Хранимые XSS возможны, когда злоумышленнику удается внедрить на сервер вредоносный код, выполняющийся в браузере каждый раз при обращении к оригинальной странице. Классическим примером этой уязвимости являются форумы, на которых разрешено оставлять комментарии в HTML-формате.

Уязвимости, вызванные кодом на стороне клиента (JavaScript, Visual Basic, Flash и т. д.).

Отраженные XSS то же самое, что и в случае с серверной стороной, только в этом случае атака возможна благодаря тому, что код обрабатывается браузером.

Хранимые XSS аналогичны хранимым XSS на стороне сервера, только в этом случае вредоносная составляющая сохраняется на клиентской стороне, используя хранилище браузера.

Уязвимости, вызванные инфраструктурой (браузер, плагины, сервера и т. д.).

Эксплуатация инфраструктурных уязвимостей на стороне клиента происходит, когда вредоносная составляющая производит какие-либо манипуляции с функциональностью браузера, например с его XSS-фильтром и т.п.

Эксплуатация инфраструктурных уязвимостей на стороне сервера происходит, когда возможно внедриться в связь между клиентом и сервером [11].

Способ защиты от XSS атак

Защита от XSS атак. Для защиты от XSS атак используют следующие методы:

— экранирование входных/выходных данных;

— использовать «белые списки»;

— установить флаг HttpOnly.

3.6 Выводы по разделу

В результате анализа атак на web-приложения были определены самые распространены и опасные атаки, такие как XSS, Brute force, CSRF, а также DoS атака. Все вышеперечисленные атаки направлены на получение чувствительной информации, кроме DoS атаки, которая направлена на достижения отказа системы. В таблице 3.2 представлена сводная информация по атакам и способам защиты от них.

Таблица 3.2 – Сводная информация по атакам и способам защиты от них

Атака	Способ защиты
XSS	Экранирование входных/выходных данных; Использовать «белые списки»; Установить флаг HttpOnly.
CSRF	Использовать XSRF токе при отправке запроса на сервер
Brute-force	максимальное количество попыток входа для одного хоста; максимальное количество попыток входа для одного пользователя; использовать капчу; удалить подсказки о неправильном вводе логина или пароля.
DoS	Производительный сервер с возможностью увеличить ресурсы; Установка и настройка брандмауэра Наличие избыточной пропускной способности, предоставленной провайдером Использование ACL; Использование балансировщиков нагрузки.

4 Разработка защищенной системы идентификации, аутентификации и авторизации

4.1 Постановка задачи

В данной главе необходимо выбрать язык программирования для разработки ЕЗСИАА, разработать архитектуру приложения, привести описание интерфейса.

4.2 Выбор языка программирования

Для реализации ЕЗСИАА можно выделить четыре основных языка программирования:

— Java;

— PHP;

— Python.

Java – это объектно-ориентированный язык программирования, приложения которого обычно транслируются в специальный байт-код, поэтому они могут работать на любой виртуальной Java-машине вне зависимости от компьютерной архитектуры [19]. Этот язык зародился как часть проекта создания передового программного обеспечения для различных бытовых приборов, т.е. был необходим платформенно независимый язык программирования, позволяющий создавать программы, которые не приходилось бы компилировать отдельно для каждой архитектуры и можно было бы использовать на различных процессорах под различными операционными системами [23].

Достоинства языка программирования Java:

— строгая статическая типизация;

— кроссплатформенность;

— автоматическое управление памятью;

— копилируемый.

PHP является скриптовым языком общего назначения, интенсивно применяемый для разработки веб-приложений. В настоящее время поддерживается подавляющим большинством хостинг-провайдеров и является одним из лидеров среди языков, применяющихся для создания динамических веб-сайтов [12].

Достоинство языка программирования PHP – это автоматическое управление памятью и кроссплатформенность.

Python - это универсальный язык высокого уровня, к преимуществам которого относят высокую производительность программных решений и структурированный, хорошо читаемый код [20]. Синтаксис данного языка максимально облегчен, что позволяет выучить его за сравнительно короткое время. Ядро имеет удобную структуру, а широкий перечень встроенных библиотек позволяет применять внушительный набор полезных функций и возможностей. Python может использоваться для написания прикладных приложений, а также разработки web-сервисов.

Достоинства языка программирования Python:

— строгая типизация

— кроссплатформенность;

— автоматическая работа с памятью.

Для разработки ЕЗСИАА был выбран язык программирования Java, так как он является строго типизированным языком и кроссплатформенным.

4.3 Описание интерфейса системы

Интерфейс ЕЗСИАА разработан на:

— HTML;

— CSS;

— JS.

Для регистрации нового клиента, владельцу ресурса, который желает встроить механизм авторизации через ЕЗСИАА, необходимо указать URL главной

страницы ресурса и его Redirect URL, затем нажать на кнопку «Зарегистрироваться». (рисунок 4.1)

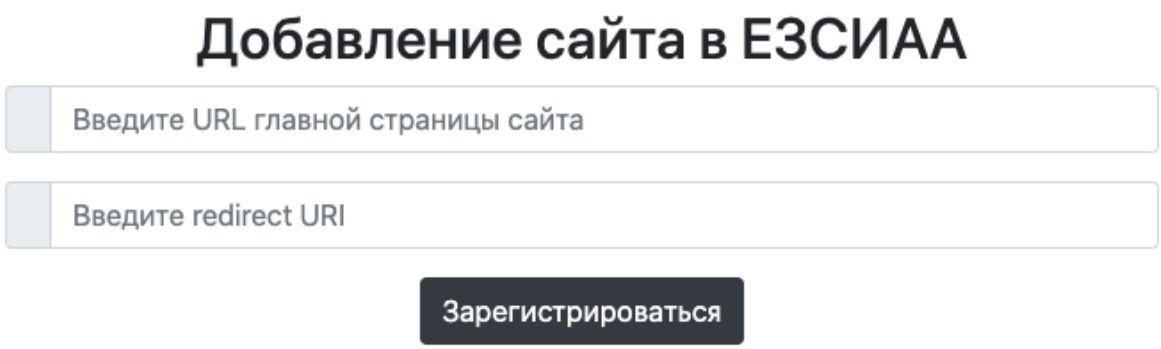

Рисунок 4.1 – Регистрация нового клиента в ЕЗСИАА

После успешной регистрации нового клиента выводится информация (рисунок 4.2):

— URL главной страницы;

— redirect URI;

— приватный идентификатор клиента;

— идентификатор клиента.

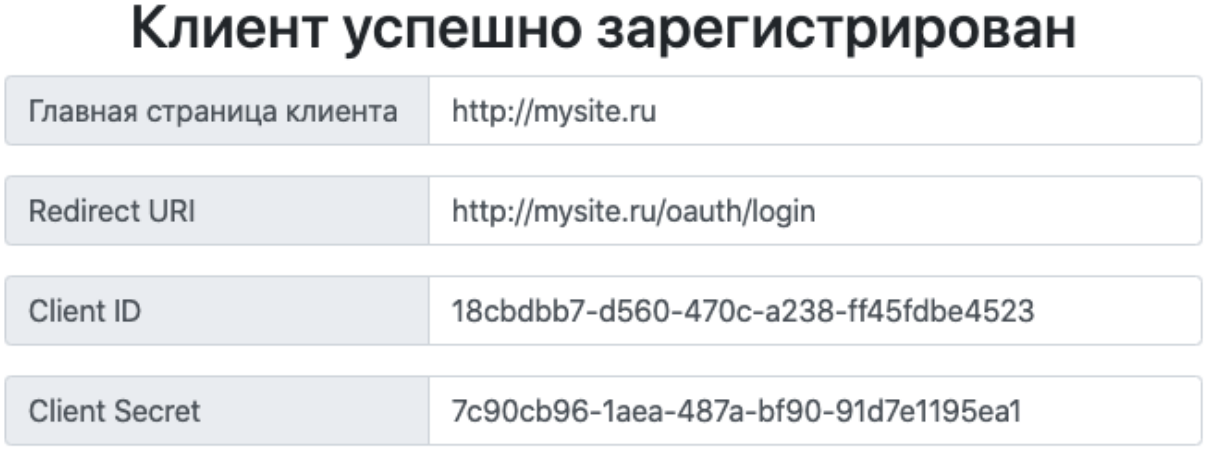

Рисунок 4.2 – Информация о новом клиенте в ЕЗСИАА

Для регистрации нового пользователя необходимо указать логин, пароль и повторить пароль, а затем нажать на кнопку «Регистрация» (рисунок 4.3).

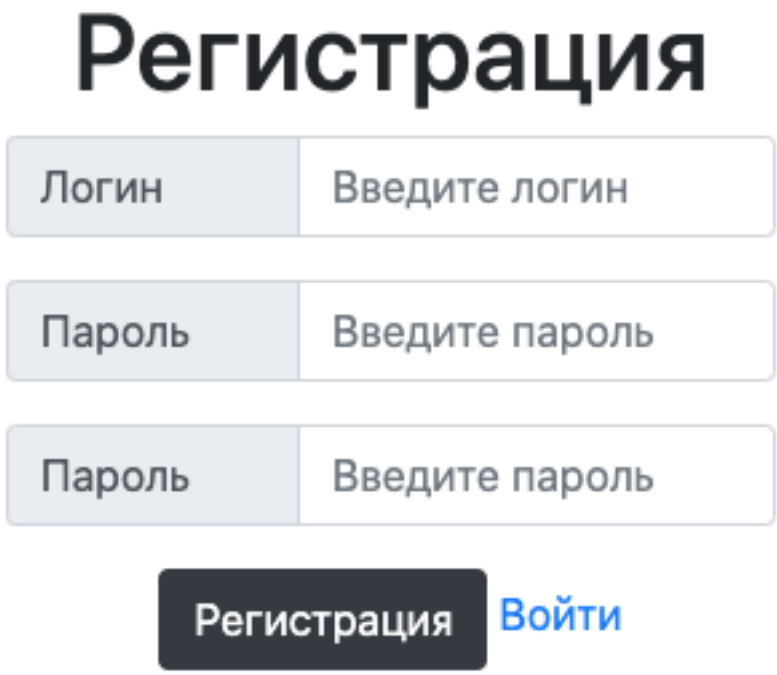

Рисунок 4.3 – Регистрация нового пользователя в ЕЗСИАА

После успешной регистрации выводится информационная табличка, сообщающая пользователю о том, что операция прошла успешно (рисунок 4.4).

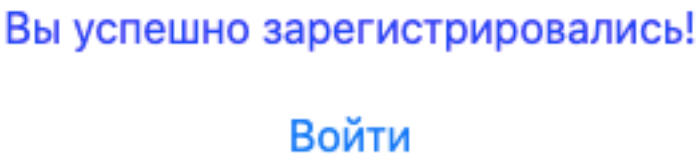

Рисунок 4.4 – Информационная табличка после успешной регистрации

Если пользователь хочет добавить информацию о себе, которую смогут получать авторизованные клиенты, то для этого ему необходимо добавить эту информацию в поля имя, город, хобби, а затем нажать на кнопку «Сохранить».

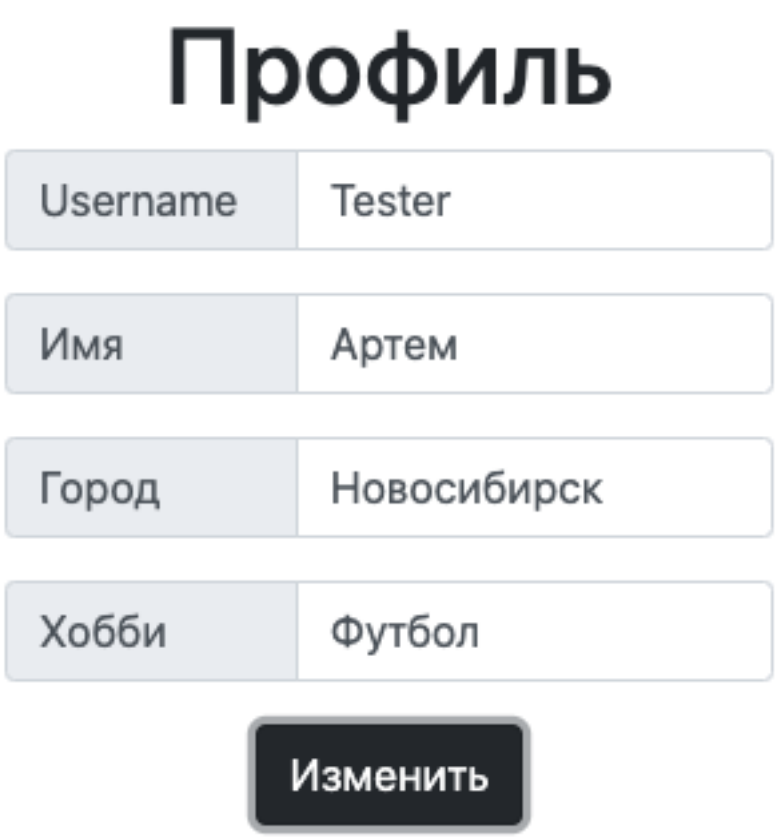

Рисунок 4.5 – Форма редактирования профиля

Для авторизации пользователя в ЕЗСИАА, ему необходимо ввести свой логин и пароль (рисунок 4.6).

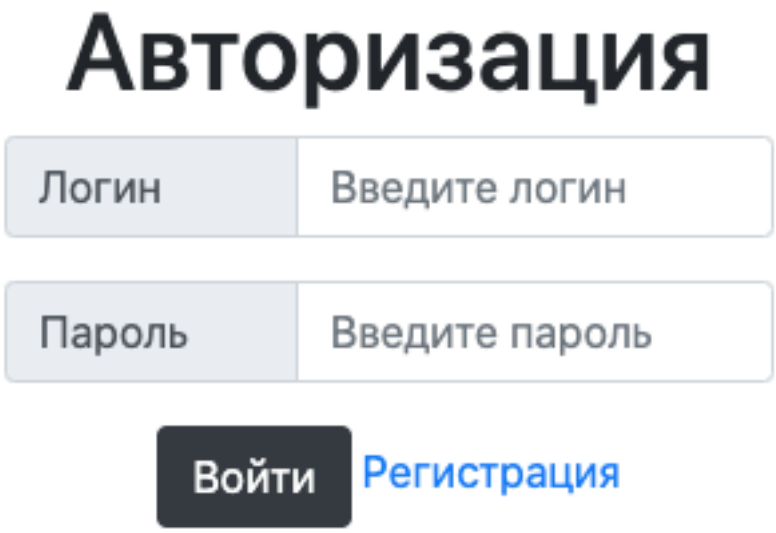

Рисунок 4.6 – Форма авторизации

Если авторизация произошла успешно, то человек попадает на главную страницу со списком всех поддерживаемых клиентов (рисунок 4.7).

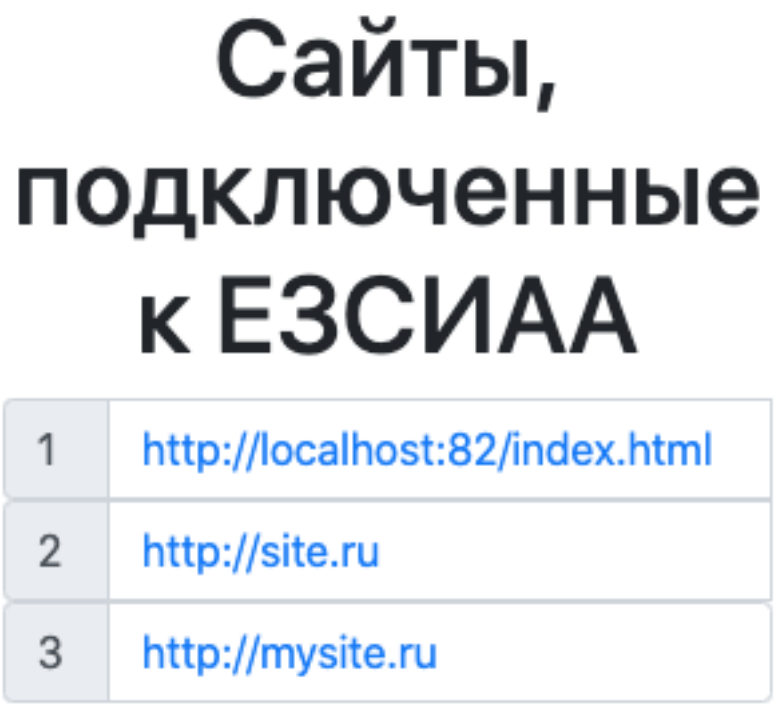

Рисунок 4.7 – Главная страница ЕЗСИАА

Для авторизации через ЕЗСИАА через внешний сервис человеку необходимо нажать на кнопку «Войта через ЕЗСИАА» (рисунок 4.8) и пройти авторизацию.

Рисунок 4.8 – Кнопка-ссылка на форму авторизации по OAuth в ЕЗСИАА

Форма авторизации по OAuth выглядит аналогично обычной форме авторизации, однако, после успешной авторизации, в отличии от обычной формы, пользователя перенаправляет на клиента, с которого была запрошена авторизация, а не на главную страницу ЕЗСИАА.

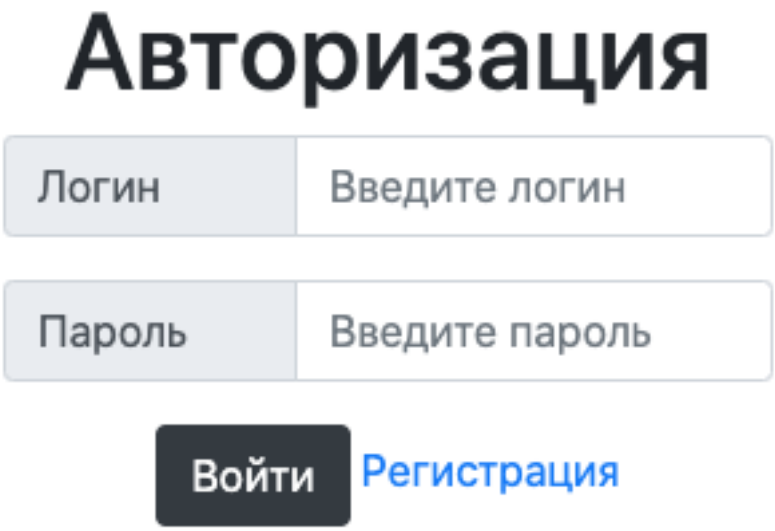

Рисунок 4.9 – Форма авторизации по OAuth2

4.4 Разработка архитектуры приложения

Для регистрации пользователя в ЕЗСИАА был разработан алгоритм, который представлен на рисунке 4.10

Рисунок 4.10 – Блок-схема алгоритма регистрации пользователя в системе

Алгоритм регистрации web-приложений(клиентов) в системе представлена на рисунке 4.11.

Рисунок 4.11 – Блок схема алгоритма регистрации web-приложений в системе

Алгоритм аутентификации пользователя в системе ЕЗСИАА представлен на рисунке 4.12.

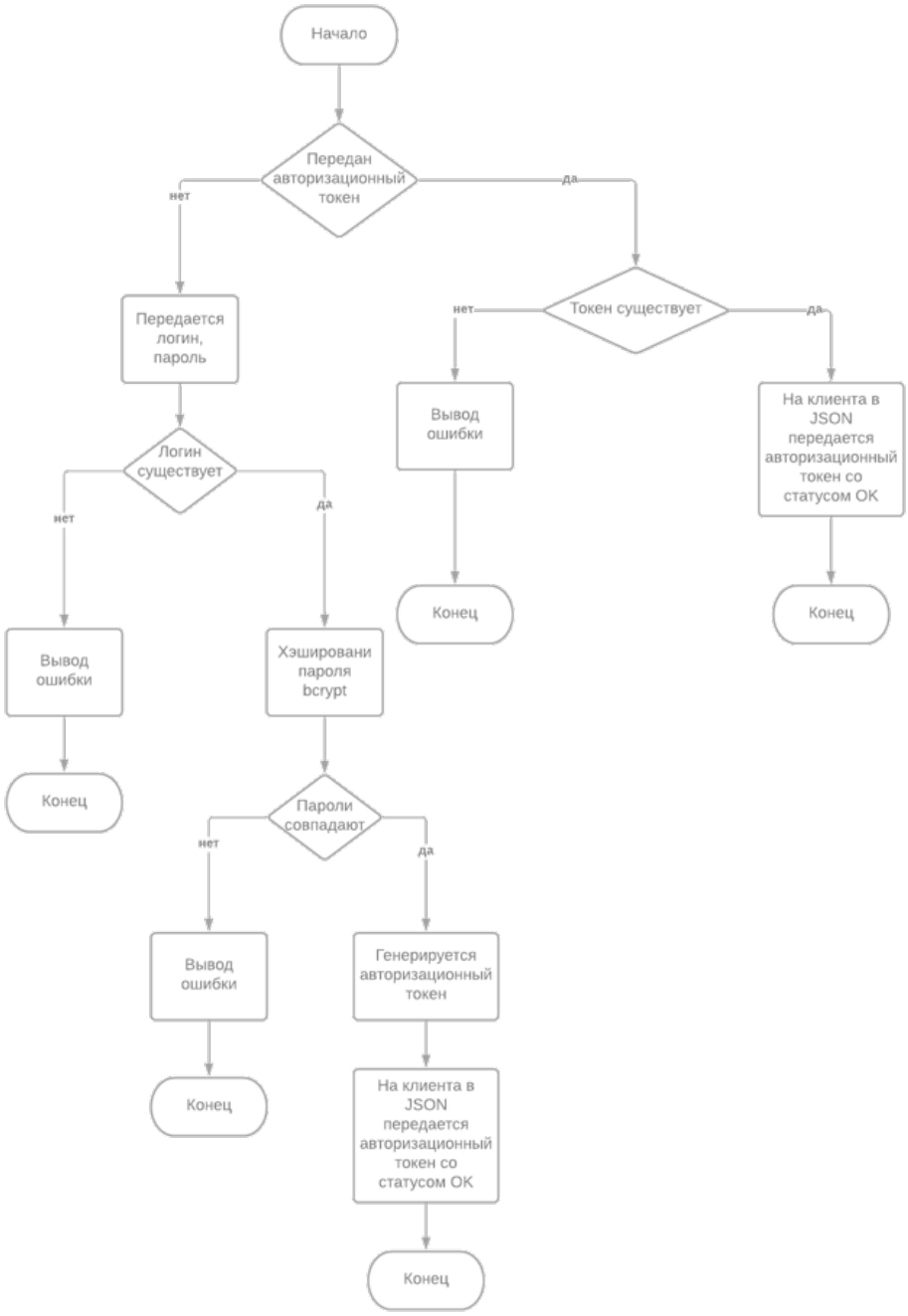

Рисунок 4.12 – Блок-схема алгоритма аутентификации пользователя в системе

Каждый пользователь, который зарегистрирован в ЕЗСИАА может изменить данные о себе. Пользователь вводит данные, которые хочет изменить в форму, которая отправляется на сервер.

Алгоритм авторизации через OIDC представлен на рисунке 4.13.

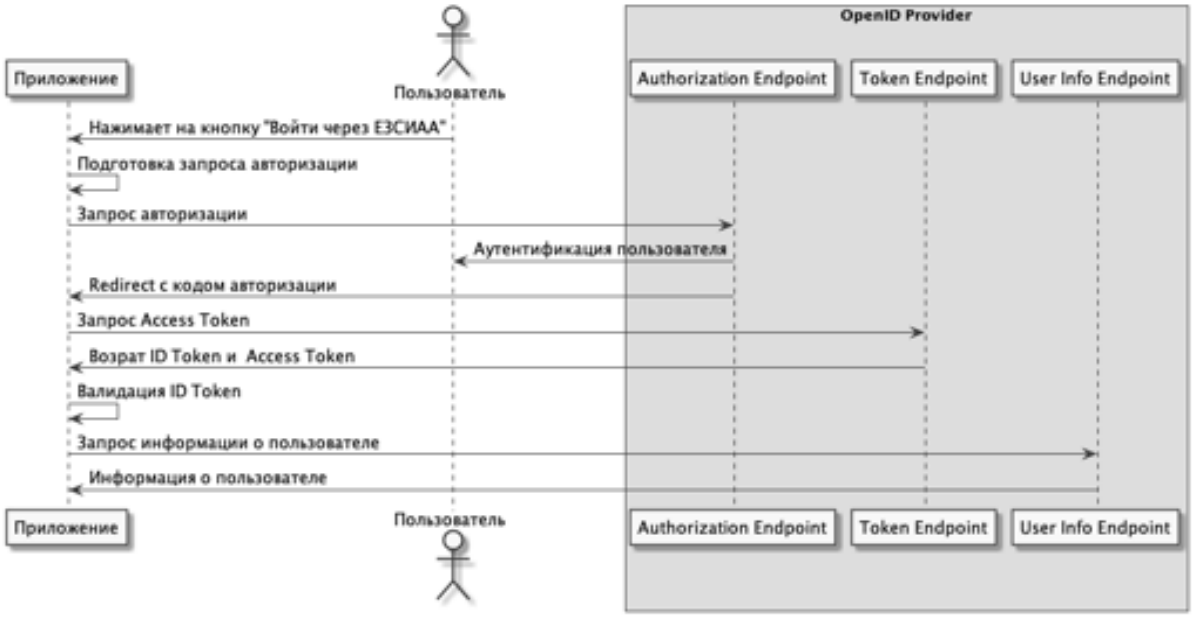

Рисунок 4.13 – Диаграмма алгоритма авторизации через OIDC

4.5 Выводы по разделу

В данной главе был выбрать язык программирования Java для разработки ЕЗСИАА, была разработана архитектура приложения и приведено описание интерфейса.

5 Безопасность жизнедеятельности

5.1 Постановка задачи

В данном разделе необходимо рассмотреть организацию охраны труда в офисе. Кроме того, необходимо раскрыть тему специальной оценки условий труда. Изучить требования к режиму труда и отдыха офисных работников и пожарную безопасность.

5.2 Организация охраны труда в офисе

Офисную работу считают одной из самых безопасных по сравнению с иными направлениями – например, работой на производстве или в строительстве. Однако сотрудники, которые проводят большую часть своего рабочего времени за компьютером, тоже подвергаются влиянию опасных факторов. Поэтому в соответствии со статьей 212 действующего Трудового кодекса РФ работодатель обязан позаботиться об охране труда офисных работников и офисной безопасности. Недостаточное внимание к комфортным условиям в офисе и охране труда не только снижает производительность труда персонала, но и может обусловить применение серьезных санкций к компании, которая не соблюдает требования трудового законодательства [13].

Требования к температуре жестко регламентированы СанПиН 2.2.4.3359-16 в среднесуточная температура за окном выше 10°С, в офисе должно быть по общему правилу 23-25°С, а если ниже этой границы – 22-24°С. Определено также, как сокращается рабочий день, если в помещении холоднее допустимого или наоборот, очень жарко [14].

Требования к организации расстановки мебели, необходимой для работы, офисной техники и других предметов внутри рабочего помещения, установлены СанПиН 2.2.2/2.4.1340-03. Обязательные к соблюдению критерии организации офисного пространства включают следующие пункты:

— площадь помещения, приходящаяся на одно рабочее место, где установлен современный компьютер с плоским монитором, должна составлять не меньше 4,5 квадратных метров. Если работник использует мониторы предыдущего поколения, оборудованные кинескопом, минимально допустимая площадь для его рабочего места должна составлять не менее 6 квадратных метров;

— рабочие места, требующие творческой концентрации или высокого умственного напряжения, должны быть изолированы перегородками высотой 1,5-2 метра;

— расстояние от глаз пользователя до экрана должно составлять 60-70 см;

— высота рабочей поверхности может регулироваться в пределах 68-80 см. Если рабочий стол не имеет опции регулировки, его поверхность нужно установить на высоте 72,5 см;

— размеры пространства для ног под столом должны иметь следующие характеристики: высота – 60 см или более, ширина – 50 см или более, глубина на уровне коленей – 45 см или более;

— установку копировально-множительной техники внутри помещения следует выполнять с соблюдением расстояния в 60 см за устройством и 100 см – перед ним.

Кроме этого, здание, где предусмотрены офисные помещения, должно иметь отдельное место для еды, характеристики которого определяются количеством работников. Так, если на предприятии трудятся менее 10 работников, для них потребуется отдельное помещение площадью не меньше 6 квадратных метров, оборудованное обеденным столом. При количестве работников до 30 человек площадь такого помещения увеличивается до 12 квадратных метров. Офис, в котором работают более 30 человек, должен иметь собственную столовую [13].

Инструктажи по охране труда являются одним из обязательных видов обучения работников. Порядок проведения и виды инструктажей по охране труда определены в Порядке обучения по охране труда и проверки знаний требований охраны труда работников организаций, утвержденном постановлением Минтруда России и Минобразования России от 13.01.2003 N 1/29

Виды инструктажей по охране труда:

— вводный инструктаж**;**

— первичный инструктаж на рабочем месте**;**

— повторный инструктаж**;**

— внеплановый инструктаж**;**

— целевой инструктаж**.**

Вводный инструктаж по охране труда проводится со всеми вновь принимаемыми на работу независимо от их образования, стажа работы по данной профессии или должности, с временными работниками, командированными, учащимися и студентами, прибывшими на производственное обучение или практику.

Первичный инструктаж по охране труда на рабочем месте проводится до начала работы руководителем подразделения или по его поручению мастером:

Повторный инструктаж по охране труда проходят все рабочие, независимо от квалификации, образования, стажа, характера выполняемой работы не реже 1 раза в 6 месяцев.

Внеплановый инструктаж по охране труда проводят при введении в действие новых или переработанных стандартов, правил, инструкций по охране труда.

Целевой инструктаж по охране труда проводят при выполнении разовых работ, не связанных с прямыми обязанностями по специальности (погрузка, выгрузка, уборка территории, разовые работы вне цеха предприятия и пр.)

5.3 Специальная оценка условий труда

Специальная оценка условий труда является единым комплексом последовательно осуществляемых мероприятий по идентификации вредных и (или) опасных факторов производственной среды и трудового процесса (далее также - вредные и (или) опасные производственные факторы) и оценке уровня их воздействия на работника с учетом отклонения их фактических значений от установленных уполномоченным Правительством Российской Федерации федеральным органом

исполнительной власти нормативов (гигиенических нормативов) условий труда и применения средств индивидуальной и коллективной защиты работников.

По результатам проведения специальной оценки условий труда устанавливаются классы условий труда на рабочих местах:

— оптимальные (1-ый класс);

— допустимые (2-ой класс);

— вредные (3-ий класс);

— опасные (4 класс).

Специальной оценке подлежат все рабочие места, которые предоставляет работодатель, будь то индивидуальный предприниматель или юридическое лицо, если трудовая деятельность организации связана непосредственно с работой машин, оборудования и прочей техники, с инструментами, с материалами и сырьем, а также с источниками, способными оказывать вредное воздействие на человека.

Специальная оценка условий труда не проводится в отношении условий труда надомников, дистанционных работников и работников, вступивших в трудовые отношения с работодателями - физическими лицами, не являющимися индивидуальными предпринимателями, или с работодателями - религиозными организациями, зарегистрированными в соответствии с федеральным законом.

Проведение специальной оценки условий труда в отношении условий труда государственных гражданских служащих и муниципальных служащих регулируется федеральными законами и иными нормативными правовыми актами Российской Федерации, законами и иными нормативными правовыми актами субъектов Российской Федерации о государственной гражданской службе и о муниципальной службе.

Вредные факторы производственной среды:

— химический;

— биологический;

— аэрозоли преимущественно фиброгенного действия;

— шум;

— инфразвук;

— ультразвук воздушный;

— вибрация общая;

— вибрация локальная;

— ионизирующее излучение;

— микроклимат;

— световая среда;

— тяжесть трудового процесса;

— напряженность трудового процесса.

5.4 Требования к режиму труда и отдыха офисных работников

Продолжительность сверхурочной работы не должна превышать для каждого работника 4 часов в течение двух дней подряд и 120 часов в год.

Согласно ст. 108 Трудового кодекса Российской Федерации на сегодняшний день, в трудовом законодательстве закреплен один вид общих перерывов в течение рабочего дня – перерыв для отдыха и питания. В течение рабочего дня (смены) работнику должен быть предоставлен перерыв для отдыха и питания продолжительностью не более двух часов и не менее 30 минут, который в рабочее время не включается. Правилами внутреннего трудового распорядка или трудовым договором может быть предусмотрено, что указанный перерыв может не предоставляться работнику, если установленная для него продолжительность ежедневной работы (смены) не превышает четырех часов.

Время предоставления перерыва и его конкретная продолжительность устанавливаются правилами внутреннего трудового распорядка или по соглашению между работником и работодателем.

На работах, где по условиям производства (работы) предоставление перерыва для отдыха и питания невозможно, работодатель обязан обеспечить работнику возможность отдыха и приема пищи в рабочее время. Перечень таких работ, а также места для отдыха и приема пищи устанавливаются правилами внутреннего трудового распорядка.

Время перерыва для отдыха и питания не включается в рабочее время и не оплачивается, поэтому работник вправе использовать его по своему усмотрению, а значит, и отлучаться на время перерыва с рабочего места, в том числе и покидать территорию работодателя. Отказ работника от своего права на использование перерыва для отдыха и питания и добровольное выполнение им своих трудовых обязанностей, не порождает обязанности работодателя оплачивать соответствующее время. Согласно ст. 258 Трудового кодекса Российской Федерации работающим женщинам, имеющим детей в возрасте до полутора лет, в том числе усыновившим ребенка либо установившим над ним опеку, предоставляются помимо перерыва для отдыха и питания дополнительные перерывы для кормления ребенка, не реже чем через каждые три часа, продолжительностью не менее тридцати минут каждый.

Каждый работник имеет право на выходные дни, то есть периоды еженедельного непрерывного отдыха. Продолжительность такого отдыха, по общему правилу, не может быть менее 42 часов.

Если человек работаете 5 дней в неделю, то ему предоставляются 2 выходных дня, обычно подряд, и общим выходным днем является воскресенье, а второй выходной день должен быть определен в правилах внутреннего трудового распорядка организации. Если определена 6-ти дневная рабочая неделя, то предоставляется 1 выходной день - воскресенье.

Продолжительность ежегодного основного оплачиваемого отпуска составляет не менее 28 календарных дней.

Согласно пункту 3.2. «ТОИ Р-45-084-01. Типовая инструкция по охране труда при работе на персональном компьютере» устанавливает, что продолжительность непрерывной работы с компьютером без регламентированного перерыва не должна превышать 2-х часов.

При продолжительности рабочей смены, вида и категории трудовой деятельности с персональным компьютером» при 8-часовой рабочей смене и работе на компьютере регламентированные перерывы следует устанавливать:

— для I категории работ через 2 часа от начала рабочей смены и через 2 часа после обеденного перерыва продолжительностью 15 минут каждый;

— для II категории работ через 2 часа от начала рабочей смены и через 1,5 - 2,0 часа после обеденного перерыва продолжительностью 15 минут каждый или продолжительностью 10 минут через каждый час работы;

— для III категории работ - через 1,5 - 2,0 часа от начала рабочей смены и через 1,5 - 2,0 часа после обеденного перерыва продолжительностью 20 минут каждый или продолжительностью 15 минут через каждый час работы [15].

5.5 Пожарная безопасность

Пожарная безопасность – состояние защищенности личности, имущества, общества и государства от пожаров.

Пожарная безопасность может быть обеспечена мерами пожарной профилактики и активной пожарной защиты.

В электроустановках причины пожаров и взрывов могут быть электрического и неэлектрического характера.

Причинами электрического характера являются:

а) искрение в электрических аппаратах и машинах, а также искрение в результате электростатических разрядов и ударов молнии;

б) токи коротких замыканий и токовые перегрузки проводников, вызывающие их перегрев до высоких температур, что может привести к воспламенению их изоляции;

в) неудовлетворительные контакты в местах соединения проводов, когда вследствие большого переходного сопротивления при протекании электрического тока выделяется значительное количество тепла и резко повышается температура контактов;

г) электрическая дуга, возникающая между контактами коммутационных аппаратов, а также при дуговой электросварке;

д) перегрузка и неисправность обмоток электрических машин и трансформаторов при отсутствии надежной защиты.

К причинам пожаров и взрывов неэлектрического характера можно отнести:

а) неосторожное обращение с огнем при проведении газосварочных работ;

б) неправильное обращение с газосварочной аппаратурой, с паяльными лампами т.д.;

в) неисправность электронагревательных приборов;

г) неисправность производственного оборудования (перегрев подшипников и т.п.), нарушение производственного технологического процесса, в результате чего возможно выделение в воздушную среду горючих газов, паров, пыли;

д) курение в пожароопасных и взрывоопасных помещениях и установках;

е) самовозгорание некоторых материалов.

В основе нормативной классификации производств по взрывной, взрывопожарной и пожарной безопасности лежат сравнительные данные, определяющие вероятность возникновения пожара или взрыва в зависимости от свойства и состояния веществ и материалов, обращающихся в производстве.

Все производства по степени взрыво - и пожарной опасности подразделяются на шесть категорий:

Категория А - особо взрыво - и пожароопасная категория. К ней отнесены производства, связанные с применением веществ, способных взрываться и гореть при взаимодействии с водой, кислородом воздуха или друг с другом; горючих газов, нижний предел воспламенения которых составляет 10% и менее, жидкостей с температурой вспышки паров до 28°С включительно при условии, что указанные газы и жидкости могут образовывать взрывоопасные смеси, превышающие 5% объема помещения (ацетон - 18°С, толуол - 4°С, растворитель № 646 - 9°С, растворитель № 648 - 13°С). К данной категории относятся нефтеперерабатывающие заводы, химические заводы, трубопроводы, склады нефтепродуктов.

К категории Б относятся производства, связанные с обращением горючих газов, нижний предел воспламенения которых более 10% к общему объему; жидкостей с температурой вспышки от 28° до 61°С включительно; жидкостей, нагре-

ваемых в условиях производства до температуры вспышки и выше; горючих пылей и волокон, нижний предел взрываемости которых 65г/м3 и менее, при условии, что указанные газы, пыли и жидкости могут образовывать взрывоопасные объемы, превышающие 5% объема помещения. К данной категории относятся цехи приготовления угольной пыли, древесной муки, сахарной пудры, выбойные и размольные отделения мельниц.

К категории В относятся производства, связанные с применением жидкостей с температурой вспышки паров выше 61°С, горючих пылей и волокон, нижний предел взрываемости которых более 65 г/м3; веществ, способных гореть при контакте с водой, кислородом воздуха или друг с другом; твердых горючих материалов. К данной категории относятся лесопильные, деревообрабатывающие, модельные, столярные цехи, склады древесных материалов.

Категория Г включает производства, связанные с применением негорючих веществ и материалов, находящихся в горячем, раскаленном или расплавленном состоянии, процесс обработки которых сопровождается выделением лучистого тепла, искр и пламени; твердых, жидких и газообразных веществ, сжигаемых или используемых в качестве топлива. Это - литейные, кузнечные, сборочно-сварочные цехи, котельные на жидком и газообразном топливе.

Категория Д включает производства, связанные с применением негорючих веществ и материалов, находящихся в холодном состоянии: слесарные, механические цехи, склады несгораемых материалов (без наличия сгораемых материалов) - чугунная чушка, арматура, столь и т.д.

В зависимости от категории производств к объектам предъявляются различные противопожарные требования. Производственные здания и сооружения в соответствии с нормами технологического проектирования подразделяются на пять категорий огнестойкости, они должны отвечать требованиям огнестойкости, т.е. фактическая степень огнестойкости здания должна быть больше требуемой [16].

Под пожарной профилактикой понимаются обучение пожарной технике безопасности и комплекс мероприятий, направленных на предупреждение пожаров. Противопожарная защита – это мероприятия, направленные на уменьшение ущерба в случае возникновения пожара.

Задачи пожарной профилактики можно разделить на три комплекса мероприятий:

— обучение, в т.ч. распространение знаний о пожаробезопасном поведении (о необходимости установки домашних индикаторов задымленности и хранения зажигалок и спичек в местах, недоступных детям);

— пожарный надзор, предусматривающий разработку государственных норм пожарной безопасности и строительных норм, а также проверку их выполнения;

— обеспечение оборудованием и технические разработки (установка переносных огнетушителей и изготовление зажигалок безопасного пользования).

Мероприятия по противопожарной защите включают:

— контроль материалов, продуктов и оборудования;

— активное ограничение распространения огня с использованием средств пожарной сигнализации, систем автоматического пожаротушения и переносных огнетушителей;

— устройство пассивных систем, ограничивающих распространение огня, дыма, жара и газов за счет секционирования помещений;

— эвакуацию людей из горящего здания в безопасное место [17].

5.6 Выводы по разделу

В данном разделе была рассмотрена организация охраны труда в офисе. Изучена и раскрыта тема специальной оценки условий труда. Проведен анализ требования к режиму труда и отдыха офисных работников. Выделены основные моменты связаны с пожарной безопасностью, а также пожарной профилактике и противопожарной безопасности.

6 Технико-экономическое обоснование работы

6.1 Постановка задачи

Цель выпускной квалификационной работы – это разработка единой защищенной системы идентификации, аутентификации и авторизации для web-приложений. Она, согласно ст. 1259 ГК РФ, относится к объектам авторских прав, таким образом, является интеллектуальной собственностью.

В данном разделе будут рассмотрены следующие вопросы:

– расчет трудоемкости и длительности работ;

– расчет себестоимости и цены программного продукта.

6.2 Составление плана по разработке программного продукта

План разработки ЕЗСИАА:

1. анализ поставленной задачи;
2. проектирование приложения;
3. разработка серверной части;
4. разработка клиентской части;
5. отладка и тестирование разработанного программного комплекса.

6.3 Расчет трудоемкости и длительности работ

Необходимо рассчитать трудоемкость и длительность работ, на основании оценки затрат времени руководителем и автором проекта, приведенным в таблице 6.1.

В этом методе для каждого этапа требуется экспертным путем определить три оценки трудоемкости, в днях:

— наименее возможная величина затрат, ai;

— наиболее вероятная величина затрат, mi;

— наиболее возможная величина затрат, bi;

На основании экспертных оценок средняя величина для ai, mi и bi определяется по формуле (6.1):

$$\overline{T} = \frac{3T_{рук}+2T_{авт}}{5}, \quad (6.1)$$

где $\overline{T}$ – среднее время, полученное на основании экспертных оценок;

$T_{рук}$ – оценка затрат времени, данная руководителем;

$T_{авт}$ – оценка затрат времени, данная автором проекта.

Величины затраченного времени будем измерять в днях.

Оценки затрат времени руководителем и автором проекта, а также расчеты средней оценки затрат времени на разработку программного продукта, в таблице 6.1

Таблица 6.1 - Время, затраченное на разработку программного продукта

Этапы разработки программного продукта	Наименее возможная величина затрат (a_i), дни			Наиболее вероятная величина затрат (m_i), дни			Наиболее возможная величина затрат (b_i), дни		
	$T_{авт}$	$T_{рук}$	$\overline{T}$	$T_{авт}$	$T_{рук}$	$\overline{T}$	$T_{авт}$	$T_{рук}$	$\overline{T}$
Анализ поставленной задачи	8	6	7,2	11	9	10,2	17	15	16,2
Проектирование приложения	14	13	13,6	18	16	17,2	21	20	20,6
Разработка серверной части	23	20	21,8	27	24	25,8	30	27	28,8
Разработка клиентской части	9	11	9,8	11	13	11,8	13	15	13,8
Отладка и тестирование разработанного программного комплекса	7	6	6,6	8	7	7,6	9	8	8,6

На основе средних оценок рассчитываются математическое ожидание и отклонение по каждому этапу разработки программного продукта. Формула расчета математического ожидания для i-го этапа:

$$MO_i = \frac{a_i + 4m_i + b_i}{6}, \quad (6.2)$$

где MO_i – математическое ожидание для i-го этапа;

a_i, m_i, b_i – средние значения.

Стандартное отклонение для каждого этапа разработки программного продукта определяется по формуле:

$$G_i = \frac{b_i - a_i}{6} \quad (6.3)$$

где Gi – стандартное отклонение по i-му этапу.

Зная математическое ожидание по каждому этапу, рассчитывается общая величина математического ожидания в целом по программному средству:

$$\text{MO} = \sum \text{MO}_\text{i} \quad (6.4)$$

где MO – общая величина математического ожидания.

Стандартное отклонение G в целом по программному средству рассчитывается по следующей формуле:

$$G = \sqrt{\sum G_i^2}, \quad (6.5)$$

где G –стандартное отклонение;

Gi – стандартное отклонение по i-му этапу.

На основе расчетов математического ожидания (6.4) и стандартного отклонения (6.5) рассчитывается коэффициент вариации – коэффициент согласованности мнения экспертов. Коэффициент вариации рассчитывается по формуле:

$$v_i = \frac{G_i}{MO_i} \quad (6.6)$$

где yi – коэффициент вариации по i-му этапу.

Все произведенные расчеты сведены в таблицу 6.2.

Таблица 6.2 – Затраты на разработку программного продукта

Этапы разработки программного продукта	Средняя величина затрат по этапам, дни			Матем. ожидание (MO_i, дни)	Станд. отклонение (G_i, дни)	Коэффициент вариации (v_i)
	Наименее возможная величина затрат (a_i, дни)	Наиболее вероятная величина затрат (m_i, дни)	Наиболее возможная величина затрат (b_i, дни)			
Анализ поставленной задачи	7,2	10,2	16,2	10,7	1,5	0,14
Проектирование приложения	13,6	17,2	20,6	17,17	1,17	0,068
Разработка серверной части	21,8	25,8	28,8	25,63	1,17	0,045
Разработка клиентской части	9,8	11,8	13,8	11,8	0,67	0,056
Отладка и тестирование разработанного программного комплекса	6,6	7,6	8,6	7,6	0,33	0,044
Итого	59	72,6	88	72,9	2,35	0,032

Из таблицы видно, что коэффициент вариации равен 0,032 и не превосходит 0,33. Поэтому мнения экспертов, исходя из данных подсчетов, считаются согласованными.

6.4 Расчет себестоимости программного продукта

Себестоимость программного продукта – это все виды затрат, понесенные при разработке продукта. Чтобы определить себестоимость разработки применяется метод экспертных оценок.

Себестоимость программного продукта определяется по формуле (6.7):

$$C = \frac{3}{m} \cdot k \cdot k_{ТЕР} \cdot k_{ПР} \cdot (t_1 + t_2) \cdot (1 + k_Н) + 8 \cdot t_3 \cdot C_М + 8 \cdot t_4 \cdot C_И, \quad (6.7)$$

$k_{\text{ТЕР}}$ – территориальный коэффициент, $k_{\text{ТЕР}} = 1{,}2$ (для НСО);

$k_{\text{ПР}}$ – коэффициент премии, $k_{\text{ПР}} = 1$;

k – коэффициент, учитывающий страховые взносы (фонды пенсионного, социального и медицинского страхования), $k = 1{,}3$;

m – количество рабочих дней в месяце, $m = 22$;

$\text{k}_\text{Н}$ – коэффициент, учитывающий накладные расходы (отопление, освещение, уборка и т. д.), $\text{k}_\text{Н} = 0{,}4$;

t_1 – время, затраченное разработчиком на анализ задачи и проектирование чел./дни;

t_2 – разработка клиентской части, разработка серверной части, время, затраченное на написание и отладку программы, чел./дни;

t_3 – время, затраченное на разработку программы с использованием машинного времени, чел./дни;

t_4 – время работы в сети интернет, дни;

$\text{С}_\text{И}$ – стоимость 1 часа работы в сети интернет, руб. (оценивается через абонентскую плату);

$\text{С}_\text{М}$ – стоимость одного часа машинного времени.

Для расчета стоимости одного часа машинного времени, необходимо определить затраты на эксплуатацию ПК за год по следующей формуле:

$$C_{\text{м}} = \frac{\text{З}_{\text{эл}} + \text{З}_{\text{а}} + \text{З}_{\text{компл}} + \text{З}_{\text{пр}}}{T_{\text{общ}}}. \quad (6.8)$$

Общее время работы компьютера за год составляет:

$$T_{\text{общ}} = 22 * 12 * 8 = 2112 \text{ (часов)}$$

Затраты на электроэнергию за год работы (на данный момент тариф $C_{\text{эл}}$ составляет 2,68 руб. за кВт/ч):

$$\text{З}_{\text{эл}} = T_{\text{общ}} * C_{\text{эл}} * P, \quad (6.9)$$

где P – потребляемая мощность ноутбука по паспортным данным в час, $P = 135$ Вт/ч.

По (6.9) затраты на электроэнергию за год работы составляют:

$$З_{эл} = 2112 * 2{,}68 * 0{,}135 = 764{,}1\ (руб.)$$

Амортизационные отчисления в год определяются как процент отчисления на амортизацию от первоначальной стоимости основных производственных фондов. Процент отчисления на амортизацию, согласно ст. 258 НК РФ, составляет 34-50% от первоначальной стоимости ПК (компьютер относится ко второй группе имущества со сроком полезного использования свыше 2 лет до 3 лет включительно). Затраты на ПК определяются по формуле:

$$З_а = С * П_р, \quad (6.10)$$

где С – стоимость ноутбука, руб.;

$П_р$ – процент отчисления на амортизацию, $П_р = 40\%$.

Получим:

$$З_а = 78000 * 0{,}4 = 31200\ (руб.)$$

Затраты на комплектующие материалы составляют:

$$З_{компл} = 5000\ (руб.)$$

Прочие расходы составляют 5% от общей суммы затрат:

$$З_{пр} = \frac{0{,}05 * (З_{эл} + З_а + З_{компл})}{0{,}95}. \quad (6.11)$$

По (7.11) прочие расходы равны:

$$З_{пр} = \frac{0{,}05 * (764{,}1 + 31200 + 5000)}{0{,}95} = 1945{,}48\ (руб.)$$

По формуле 5.8 стоимость одного часа машинного времени равна:

$$C_м = \frac{764{,}1 + 31200 + 5000 + 1945{,}48}{2112} = 18{,}4231\ (руб.)$$

Тариф на услугу интернет составляет 635 руб. в месяц, следовательно, стоимость 1 часа работы в сети интернет равен:

$$С_и = \frac{635}{30} = 21{,}2\ (руб.)$$

Заключительным этапом расчета является распределение ранее рассчитанной трудоемкости (таблица 6.2) по 4 направлениям:

– t_1 включает первые 2 этапа:

$$t_1 = 10{,}7 + 17{,}17 = 27{,}87 \text{ (дней)}$$

– t_2 включает оставшиеся этапе:

$$t_2 = 25{,}63 + 11{,}8 + 7{,}6 = 45{,}03 \text{ (дней)}$$

– t_3 включает время работы ПК для разработки программы:

$$t_3 = 75 \text{ (дней)}$$

– t_4 включает время использования интернета для разработки программы:

$$t_4 = 70{,}5 \text{ (дней)}$$

Наконец, итоговая себестоимость программного продукта составляет:

$$\text{С} = \frac{40000}{22} \cdot 1{,}3 \cdot 1{,}2 \cdot 1 \cdot (27{,}87 + 45{,}03) \cdot (1 + 0{,}4) + 8 \cdot 75 \cdot 18{,}4231 + 8 \cdot 70{,}5 \cdot 21{,}2 = 312489{,}94 \text{ (руб.)}$$

В случае, если программный продукт будет доработан и реализован на рынке, следует рассчитать цену по следующей формуле:

$$\text{Ц} = C * (1 + \frac{P}{100}), \quad (6.12)$$

где C – себестоимость разработки программы, руб;

P – рентабельность, руб.

Далее определяется цена программного продукта, при условии, что значение рентабельности равно 20%:

$$\text{Ц} = 312489{,}94 \cdot \left(1 + \frac{20}{100}\right) = 374987{,}9 \text{ (руб.)}$$

Цена с учетом налога на добавленную стоимость находится по формуле:

$$\text{Ц}_{\text{НДС}} = \text{Ц} * \text{К}_{\text{НДС}}, \quad (6.13)$$

где Ц – цена программного продукта;

$\text{К}_{\text{НДС}}$ – коэффициент, учитывающий ставку налога на добавленную стоимость (НДС), $\text{К}_{\text{НДС}} = 1{,}20$.

Цена с учетом налога на добавленную стоимость составит:

$$\text{Ц}_{\text{НДС}} = 374987{,}9 * 1{,}20 = 449985{,}48 \text{ (руб.)}$$

6.5 Выводы по разделу

В данном разделе была определены и рассчитаны трудоемкость и длительность работ, а также рассчитаны себестоимость и цена программного продукта.

По итогам расчетов себестоимость разработки программного продукта составила 312489,94 рублей. В случае, если программный продукт будет доработан и будет реализован на рынке, то стоимость продукта с учетом рентабельности в 20% и налога НДС 20% составит 449985,48 рублей. Данная цена сопоставима с лидерами рынка (VK, Facebook, Google), которые предоставляют функциональность авторизации пользователя 3ей стороной, из-за того, что система является программным кодом и требует примерно равных ресурсов, поэтому можно сделать вывод о том, что разработка данного продукта является целесообразной.

Заключение

В данной работе была спроектирована единая защищенная система идентификации, аутентификации и авторизации для web-приложений, проанализирован протокол OAuth и возможные атаки на авторизационный сервер, а затем разработана единая защищенная система идентификации, аутентификации и авторизации для web-приложений.

В рамках данной работы были решены следующие задачи:

— проектирование единой защищенной системы идентификации, аутентификации и авторизации для web-приложений;

— анализ протокола OAuth2;

— анализ атак на авторизационный сервер;

— разработка защищенной системы идентификации, аутентификации и авторизации;

— рассмотрены вопросы безопасности жизнедеятельности;

— выполнены технико-экономическое расчеты.

Список литературы

1 Обзор способов и протоколов аутентификации в веб-приложениях URL: https://habr.com/ru/company/dataart/blog/262817/ (дата обращения: 11.10.2020)

2 Укрощаем протоколы доверия – OAuth авторизация с InterSystems URL: https://habr.com/ru/company/intersystems/blog/466699/ (дата обращения: 11.10.2020)

3 Введение в OAuth 2 URL: https://www.digitalocean.com/community/tutorials/oauth-2-ru (дата обращения: 13.10.2020)

4 Атака CSRF URL: https://learn.javascript.ru/csrf (дата обращения: 07.01.2021)

5 Атаки на веб-приложения/ итоги 2018 года URL: https://www.ptsecurity.com/ru-ru/research/analytics/web-application-attacks-2019/ (дата обращения: 05.11.2020)

6 Методы защиты от CSRF-атаки URL: https://habr.com/ru/post/318748/ (дата обращения: 08.11.2020)

7 Брутфорс (Brute force) URL: https://www.anti-malware.ru/threats/brute-force (дата обращения: 15.11.2020)

8 Распределенные сетевые атаки / DDoS URL: https://www.kaspersky.ru/resource-center/threats/ddos-attacks (дата обращения: 15.11.2020)

9 DDoS–атаки. Причины возникновения, классификация и защита от DDoS-атак URL: https://efsol.ru/articles/ddos-attacks.html (дата обращения: 22.11.2020)

10 Что такое DDoS-атаки URL: https://aws.amazon.com/ru/shield/ddos-attack-protection/ (дата обращения: 22.11.2020)

11 Что собой представляет XSS-уязвимость URL: https://www.anti-malware.ru/what-is-an-xss-vulnerability (дата обращения: 22.11.2020)

12 Что такое PHP URL: http://web.spt42.ru/index.php/chto-takoe-php (дата обращения: 03.12.2020)

13 Охрана труда в офисе - инструкция для офисных работников URL: https://oxrana-bez.ru/stati/ohrana-truda-dlya-ofisnyh-rabotnikov/ (дата обращения: 03.12.2020)

14 СанПиН для офисных работников URL http://www.garant.ru/infografika/1093322/ (дата обращения: 07.12.2020)

15 Технические перерывы URL: https://онлайнинспекция.рф/questions/view/111160 (дата обращения: 10.12.2020)

16 Меры противопожарной профилактики URL: http://adminnovod.ru/index.php?katalog=gochs&id=921 (дата обращения: 17.12.2020)

17 Пожарная профилактика и противопожарная защита URL: https://zmsk.mos.ru/law-and-order/su-mch/fire-prevention-and-fire-protection.php (дата обращения: 17.12.2020)

18 Системы единого входа (Single Sign-On, SSO) URL: https://www.anti-malware.ru/security/single-sign-on (дата обращения: 21.12.2020)

19 Технологии URL: https://solid-s.ru/langs.html (дата обращения: 25.12.2020)

20 Python URL: https://bizzapps.ru/p/python/ (дата обращения: 25.12.2020)

21 Brute-force атаки с использованием Kali Linux URL: https://habr.com/ru/company/pentestit/blog/434216/ (дата обращения: 07.01.2021)

22 Web Application Firewall (WAF) URL: https://www.anti-malware.ru/security/web-application-firewall (дата обращения: 07.01.2021)

23 Java URL: http://progopedia.ru/language/java/ (дата обращения: 07.01.2021)

24 Differences Between OAuth 1 and 2 URL: https://www.oauth.com/oauth2-servers/differences-between-oauth-1-2/ (дата обращения: 07.01.2021)

25 RFC 7636 - Proof Key for Code Exchange by OAuth Public Clients https://tools.ietf.org/html/rfc7636#section-6.2 (дата обращения: 25.10.2020)

Приложение А

Исходный код программы

add_client.html

```
<!doctype html>
<html lang="en">
<head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-
fit=no">

    <!-- Bootstrap CSS -->
    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/css/bootstrap.min.css"
          integrity="sha384-
TX8t27EcRE3e/ihU7zmQxVncDAy5uIKz4rEkgIXeMed4M0jlfIDPvg6uqKI2xXr2" cros-
sorigin="anonymous">

    <title>Добавление клиента</title>
</head>
<body>
<nav class="navbar navbar-expand-lg navbar navbar navbar-dark bg-dark">
    <a class="navbar-brand" href="./index.html">ЕЗСИАА</a>
    <button class="navbar-toggler" type="button" data-toggle="collapse" data-
target="#navbarSupportedContent"
            aria-controls="navbarSupportedContent" aria-expanded="false" aria-
label="Toggle navigation">
        <span class="navbar-toggler-icon"></span>
    </button>
</nav>
<div class="layer-left" id="hide-div">
    <h2>Добавление сайта в ЕЗСИАА</h2>
    <div class="input-group mb-3">
        <div class="input-group-prepend">
            <span class="input-group-text" id="basic-addon1"></span>
        </div>
        <input id="nameClient" type="text" class="form-control" placehold-
er="Введите URL главной страницы сайта"
               aria-label="Username" aria-describedby="basic-addon1">
    </div>
    <div class="input-group mb-3">
        <div class="input-group-prepend">
            <span class="input-group-text" id="basic-addon1"></span>
        </div>
        <input id="urlClient" type="text" class="form-control" placehold-
er="Введите redirect URI" aria-label="Username"
               aria-describedby="basic-addon1">
    </div>
    <button type="submit" onclick="f()" class="btn btn-
dark">Зарегистрироваться</button>
</div>
<div class="layer-left" id="hide-dev-second" style="display: none">
    <h2>Клиент успешно зарегистрирован</h2>
    <div class="input-group mb-3">
        <div class="input-group-prepend">
```

```
            <span class="input-group-text group-text-size" id="basic-
addon2">Главная страница клиента</span>
        </div>
        <span id="nameClientResult" class="form-control result-align" aria-
label="Username" aria-describedby="basic-addon1"></span>
    </div>
    <div class="input-group mb-3">
        <div class="input-group-prepend">
            <span class="input-group-text group-text-size" id="basic-
addon3">Redirect URI</span>
        </div>
        <span id="redirectUrl" class="form-control result-align" aria-
label="Username" aria-describedby="basic-addon1"></span>
    </div>
    <div class="input-group mb-3">
        <div class="input-group-prepend">
            <span class="input-group-text group-text-size" id="basic-
addon4">Client ID</span>
        </div>
        <span id="clientId" class="form-control result-align" aria-
label="Username" aria-describedby="basic-addon1"></span>
    </div>
    <div class="input-group mb-3">
        <div class="input-group-prepend">
            <span class="input-group-text group-text-size" id="basic-
addon5">Client Secret</span>
        </div>
        <span id="clientSecret" class="form-control result-align" aria-
label="Username" aria-describedby="basic-addon1"></span>
    </div>
</div>
<script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"
        integrity="sha384-
DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkfj"
        crossorigin="anonymous"></script>
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/js/bootstrap.bundle.min.js"
        integrity="sha384-
ho+j7jyWK8fNQe+A12Hb8AhRq26LrZ/JpcUGGOn+Y7RsweNrtN/tE3MoK7ZeZDyx"
        crossorigin="anonymous"></script>
</body>
<script>
    function f() {
        const url = 'http://127.0.0.1:8443/client/create';

        const nameClient = es-
capeHtml(document.getElementById("nameClient").value);
        const urlClient = escapeHtml(document.getElementById("urlClient").value);

        try {
            const response = fetch(url, {
                method: 'POST',
                credentials: 'include',
                body: JSON.stringify({
                    name: nameClient,
                    redirectUri: urlClient
                }),
                headers: {
                    'Content-Type': 'application/json'
                }
            })
                .then(response => response.json())
                .then(json => {
                    console.log('Успех:', JSON.stringify(json));
```

```
                    var x = document.getElementById("hide-div");
                    x.style.display = "none";
                    var y = document.getElementById("hide-dev-second");
                    y.style.display = "block";
                    document.getElementById("nameClientResult").innerHTML = name-
Client;
                    document.getElementById("redirectUrl").innerHTML = urlClient;
                    document.getElementById("clientId").innerHTML =
json["data"].clientId;
                    document.getElementById("clientSecret").innerHTML =
json["data"].clientSecret;
                });
        } catch (error) {
            console.error('Ошибка:', error);
        }
    }

    function escapeHtml(unsafe) {
        return unsafe
            .replace(/&/g, "&amp;")
            .replace(/</g, "&lt;")
            .replace(/>/g, "&gt;")
            .replace(/"/g, "&quot;")
            .replace(/'/g, "&#039;");
    }
</script>
<style>
    .result-align {
        text-align: left;
    }
    .layer-left {
        text-align: center;
        width: 45%;
        margin: auto;
        margin-top: 50px;

    }
    .group-text-size {
        width: 230px;
    }
</style>
</html>
```

edit_client.html

```
<!doctype html>
<html lang="en">
<script>
    function unescapeHTML(escapedHTML) {
        return escapedHTML.replace(/&lt;/g, '<').replace(/&gt;/g,
'>').replace(/&amp;/g, '&');
    }

    const token = localStorage.getItem('token');
    if (token === "" || token == null) {
        window.location.href = './login.html'
    } else {
        try {
            fetch('http://127.0.0.1:8443/session', {
                method: 'POST',
                headers: {
                    'Content-type': 'application/json; charset=UTF-8',
```

```
                    'Authorization': localStorage.getItem('token'),
                    'X-CSRF-TOKEN': localStorage.getItem("csrfToken")
                },
                body: JSON.stringify({
                    sessionId: token
                }),
            })
                .then(response => response.json())
                .then(json => {
                    if (json.status !== "OK") {
                        window.location.href = './login.html';
                        localStorage.clear();
                    }
                    console.log(json);
                    fetch('http://127.0.0.1:8443/user/resources', {
                        method: 'POST',
                        headers: {
                            'Content-type': 'application/json; charset=UTF-8',
                            'Authorization': localStorage.getItem('token'),
                            'X-CSRF-TOKEN': localStorage.getItem("csrfToken")
                        }
                    })
                        .then(response => response.json())
                        .then(json => {
                            console.log(json);
                            document.getElementById("usernameProfile").innerHTML =
unescapeHTML(json["data"].username);
                            document.getElementById("nameProfile").value = unes-
capeHTML(json["data"].name);
                            document.getElementById("cityProfile").value = unes-
capeHTML(json["data"].city);
                            document.getElementById("hobbyProfile").value = unes-
capeHTML(json["data"].hobby);
                        });
                });
        } catch (error) {
            console.log("Ошибка", error);
            localStorage.clear();
            window.location.href = "./login.html";
        }
    }
</script>
<head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-
fit=no">

    <!-- Bootstrap CSS -->
    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/css/bootstrap.min.css"
          integrity="sha384-
TX8t27EcRE3e/ihU7zmQxVncDAy5uIKz4rEkgIXeMed4M0jlfIDPvg6uqKI2xXr2" cros-
sorigin="anonymous">

    <title>Профиль</title>
</head>
<body>
<nav class="navbar navbar-expand-lg navbar navbar navbar-dark bg-dark">
    <a class="navbar-brand" href="index.html">ЕЗСИАА</a>
    <button class="navbar-toggler" type="button" data-toggle="collapse" data-
target="#navbarSupportedContent"
            aria-controls="navbarSupportedContent" aria-expanded="false" aria-
label="Toggle navigation">
```

```
        <span class="navbar-toggler-icon"></span>
    </button>

    <div class="collapse navbar-collapse" id="navbarSupportedContent">
        <ul class="navbar-nav mr-auto">
            <li class="nav-item active">
                <a class="nav-link" href="edit_client.html">Профиль<span
class="sr-only">(current)</span></a>
            </li>
        </ul>
        <button class="btn btn-outline-success " onclick="logout()"
type="submit">Выйти</button>
    </div>
</nav>
<div id="hidden-div-profile" class="layer-left">
    <h1>Профиль</h1>
    <div class="input-group mb-3">
        <div class="input-group-prepend">
            <span class="input-group-text group-text-size" id="basic-
addon3">Username</span>
        </div>
        <span id="usernameProfile" class="form-control result-align" aria-
label="Username" aria-describedby="basic-addon1"></span>
    </div>

    <div class="input-group mb-3">
        <div class="input-group-prepend">
            <span class="input-group-text group-text-size" id="basic-
addon3">Имя</span>
        </div>
        <input id="nameProfile" placeholder="Введите имя" class="form-control re-
sult-align" aria-label="Username" aria-describedby="basic-addon1">
    </div>

    <div class="input-group mb-3">
        <div class="input-group-prepend">
            <span class="input-group-text group-text-size" id="basic-
addon3">Город</span>
        </div>
        <input id="cityProfile" placeholder="Введите город" class="form-control
result-align" aria-label="Username" aria-describedby="basic-addon1">
    </div>

    <div class="input-group mb-3">
        <div class="input-group-prepend">
            <span class="input-group-text group-text-size" id="basic-
addon3">Хобби</span>
        </div>
        <input id="hobbyProfile" placeholder="Введите хобби" class="form-control
result-align" aria-label="Username" aria-describedby="basic-addon1">
    </div>
    <button type="submit" onclick="f()" class="btn btn-dark" data-toggle="modal"
data-target="#exampleModal">Изменить</button>

    <div class="modal fade" id="exampleModal" tabindex="-1" role="dialog" aria-
labelledby="exampleModalLabel" aria-hidden="true">
        <div class="modal-dialog" role="document">
            <div class="modal-content">
                <div class="modal-header">
                    <h5 class="modal-title" id="exampleModalLabel">Профиль</h5>
                    <button type="button" class="close" data-dismiss="modal" aria-
label="Close">
                        <span aria-hidden="true">&times;</span>
                    </button>
```

```
                </div>
                <div class="modal-body">
                    Изменения сохранены
                </div>
                <div class="modal-footer">
                    <button type="button" class="btn btn-secondary" data-
dismiss="modal">Close</button>
                </div>
            </div>
        </div>
    </div>



</div>
<script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"
        integrity="sha384-
DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkfj"
        crossorigin="anonymous"></script>
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/js/bootstrap.bundle.min.js"
        integrity="sha384-
ho+j7jyWK8fNQe+A12Hb8AhRq26LrZ/JpcUGGOn+Y7RsweNrtN/tE3MoK7ZeZDyx"
        crossorigin="anonymous"></script>
</body>
<script>
    function f() {
        const url = 'http://127.0.0.1:8443/user/change';

        const nameEdit = escapeHtml(document.getElementById("nameProfile").value);
        const hobbyEdit = es-
capeHtml(document.getElementById("hobbyProfile").value);
        const cityEdit = escapeHtml(document.getElementById("cityProfile").value);

        try {
            fetch(url, {
                method: 'POST',
                credentials: 'include',
                body: JSON.stringify({
                    name: nameEdit,
                    city: cityEdit,
                    hobby: hobbyEdit
                }),
                headers: {
                    'Content-Type': 'application/json',
                    'Authorization': localStorage.getItem('token'),
                    'X-CSRF-TOKEN': localStorage.getItem("csrfToken")
                }
            })
                .then(response => response.json())
                .then(json => {
                    console.log('Успех:', JSON.stringify(json));
                });
        } catch (error) {
            console.error('Ошибка:', error);
            localStorage.clear();
            window.location.href = "./login.html";
        }
    }

    function escapeHtml(unsafe) {
        return unsafe
            .replace(/&/g, "&amp;")
            .replace(/</g, "&lt;")
```

```
            .replace(/>/g, "&gt;")
            .replace(/"/g, "&quot;")
            .replace(/'/g, "&#039;");
    }

    function g() {
        var x = document.getElementById("hidden-edit");
        x.style.display = "block";
        var y = document.getElementById("hidden-div-profile");
        y.style.display = "none";
    }

    function logout() {
        fetch('http://127.0.0.1:8443/logout', {
            method: 'GET',
            headers: {
                'Authorization': localStorage.getItem('token'),
                'X-CSRF-TOKEN': localStorage.getItem("csrfToken")
            },
        })
            .then(response => response.json())
            .then(json => {
                console.log('Успех:', JSON.stringify(json));
                localStorage.clear();
                window.location.href = './login.html'
            });
    }
</script>
<style>
    .result-align {
        text-align: left;
    }
    .layer-left {
        text-align: center;
        width: 20%;
        margin: auto;
        margin-top: 50px;
    }
    .group-text-size {
        width: 110px;
    }
</style>
</html>
```

index.html

```
<!doctype html>
<html lang="en">
<script>
    const token = localStorage.getItem('token');
    if (token === "" || token == null) {
        window.location.href = './login.html'
    } else {
        try {
            fetch('http://127.0.0.1:8443/session', {
                method: 'POST',
                headers: {
                    'Content-type': 'application/json; charset=UTF-8',
                    'Authorization': localStorage.getItem('token'),
                    'X-CSRF-TOKEN': localStorage.getItem("csrfToken")
```

```
                },
            })
                .then(response => response.json())
                .then(json => {
                    if (json.status !== "OK") {
                        window.location.href = './login.html';
                        localStorage.clear();
                    }
                    console.log(json);

                    fetch('http://127.0.0.1:8443/client/fetch/all', {
                        method: 'POST',
                        headers: {
                            'Content-type': 'application/json; charset=UTF-8',
                            'Authorization': localStorage.getItem('token'),
                            'X-CSRF-TOKEN': localStorage.getItem("csrfToken")
                        }
                    })

                        .then(response => response.json())
                        .then(json => {
                            console.log(json);
                            let worker = json["data"].clientList;
                            let out_arr = document.getElementById('out_arr');
                            let str = ' ';
                            for (let i = 0; i < worker.length; i++) {
                                if (worker[i] !== undefined) {
                                    str += '<div class="input-group mb-3"> <div
class="input-group-prepend"><span class="input-group-text group-text-size"
id="basic-addon3">' + (i+1) + '</span></div>'
                                    str += '<span class="form-control result-
align"><a href="' + worker[i] + '">' + worker[i] + '</a></span>';
                                }
                            }
                            out_arr.innerHTML = str;
                        });
                });
        } catch (error) {
            console.log("Ошибка", error);
            localStorage.clear();
            window.location.href = "./login.html";
        }
    }
</script>
<head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-
fit=no">

    <!-- Bootstrap CSS -->
    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/css/bootstrap.min.css"
          integrity="sha384-
TX8t27EcRE3e/ihU7zmQxVncDAy5uIKz4rEkgIXeMed4M0jlfIDPvg6uqKI2xXr2" cros-
sorigin="anonymous">

    <title>ЕЗСИАА</title>
</head>
<body>
<nav class="navbar navbar-expand-lg navbar navbar navbar-dark bg-dark">
    <a class="navbar-brand" href="#">ЕЗСИАА</a>
    <button class="navbar-toggler" type="button" data-toggle="collapse" data-
target="#navbarSupportedContent"
```

```
            aria-controls="navbarSupportedContent" aria-expanded="false" aria-
label="Toggle navigation">
        <span class="navbar-toggler-icon"></span>
    </button>

    <div class="collapse navbar-collapse" id="navbarSupportedContent">
        <ul class="navbar-nav mr-auto">
            <li class="nav-item active">
                <a class="nav-link" href="edit_client.html">Профиль<span
class="sr-only">(current)</span></a>
            </li>
        </ul>
        <button class="btn btn-outline-success " onclick="logout()"
type="submit">Выйти</button>
    </div>
</nav>
<div id="hidden-div-profile" class="layer-left">
    <h1>Сайты, подключенные к ЕЗСИАА</h1>

    <span id="out_arr"></span>
    <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"
            integrity="sha384-
DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkfj"
            crossorigin="anonymous"></script>
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/js/bootstrap.bundle.min.js"
            integrity="sha384-
ho+j7jyWK8fNQe+A12Hb8AhRq26LrZ/JpcUGGOn+Y7RsweNrtN/tE3MoK7ZeZDyx"
            crossorigin="anonymous"></script>
</body>
<script>
    function logout() {
        fetch('http://127.0.0.1:8443/logout', {
            method: 'GET',
            headers: {
                'Authorization': localStorage.getItem('token'),
                'X-CSRF-TOKEN': localStorage.getItem("csrfToken")
            },
        })
            .then(response => response.json())
            .then(json => {
                console.log('Успех:', JSON.stringify(json));
                localStorage.clear();
                window.location.href = './login.html'
            });
    }
</script>
<style>
    .result-align {
        text-align: left;
    }
    .layer-left {
        text-align: center;
        width: 20%;
        margin: auto;
        margin-top: 50px;

    }
    .group-text-size {
        width: 40px;
    }
</style>
</html>
```

login_oauth.html

```
<!doctype html>
<html lang="en">
<script>
    function get(name) {
        if (name = (new RegExp('[?&]' + encodeURIComponent(name) +
'=([^&]*)')).exec(location.search))
            return decodeURIComponent(name[1]);
    }

    const token = localStorage.getItem('token');
    if (token !== "" && token != null) {
        try {
            fetch('http://127.0.0.1:8443/session/oauth' + window.location.search,
{
                method: 'POST',
                headers: {
                    'Content-type': 'application/json; charset=UTF-8',
                    'Authorization': localStorage.getItem('token'),
                    'X-CSRF-TOKEN': localStorage.getItem("csrfToken")
                },
            })
                .then(response => response.json())
                .then(json => {
                    console.log("status", json["status"]);
                    if (json["status"] === "OK") {
                        var redirectUriParsed = get("redirect_uri");
                        console.log("redirect_uri", redirectUriParsed);
                        var los = redirectUriParsed + "?code=" +
json["data"].code;
                        window.location.href = redirectUriParsed + "?code=" +
json["data"].code;
                    } else {
                        localStorage.clear();
                    }
                    console.log(json);
                });
        } catch (error) {
            console.log("Ошибка", error);
            localStorage.clear();
        }
    }
</script>
<head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-
fit=no">

    <!-- Bootstrap CSS -->
    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/css/bootstrap.min.css"
          integrity="sha384-
TX8t27EcRE3e/ihU7zmQxVncDAy5uIKz4rEkgIXeMed4M0jlfIDPvg6uqKI2xXr2" cros-
sorigin="anonymous">

    <title>Авторизация</title>
</head>
```

```
<body>
<nav class="navbar navbar-expand-lg navbar navbar navbar-dark bg-dark">
    <a class="navbar-brand" href="#">ЕЗСИАА</a>
</nav>
<div id="hidden-div-profile" class="layer-left">
    <h1>Авторизация</h1>
    <span id="error-message" style="display: none">Неверный логин или
пароль</span>
    <div class="input-group mb-3">
        <div class="input-group-prepend">
            <span class="input-group-text group-text-size" id="basic-
addon3">Логин</span>
        </div>
        <input id="loginSingIn" placeholder="Введите логин" class="form-control
result-align" aria-label="Username"
               aria-describedby="basic-addon1">
    </div>

    <div class="input-group mb-3">
        <div class="input-group-prepend">
            <span class="input-group-text group-text-size" id="basic-
addon3">Пароль</span>
        </div>
        <input type="password" id="passwordSingIn" placeholder="Введите пароль"
class="form-control result-align"
               aria-label="Username" aria-describedby="basic-addon1">
    </div>
    <button type="submit" onclick="f()" class="btn btn-dark">Войти</button>
    <a href="register.html">Регистрация</a>
</div>
<script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"
        integrity="sha384-
DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkfj"
        crossorigin="anonymous"></script>
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/js/bootstrap.bundle.min.js"
        integrity="sha384-
ho+j7jyWK8fNQe+A12Hb8AhRq26LrZ/JpcUGGOn+Y7RsweNrtN/tE3MoK7ZeZDyx"
        crossorigin="anonymous"></script>
</body>
<script>
    function f() {
        const url = 'http://127.0.0.1:8443/oauth/login' + window.location.search;

        const loginSingIn = es-
capeHtml(document.getElementById("loginSingIn").value);
        const passwordSingIn = es-
capeHtml(document.getElementById("passwordSingIn").value);

        try {
            fetch(url, {
                method: 'POST',
                credentials: 'include',
                body: JSON.stringify({
                    username: loginSingIn,
                    password: passwordSingIn
                }),
                headers: {
                    'Content-Type': 'application/json',
                    'Origin': 'http://localhost:81'
                }
            })
                .then(response => response.json())
                .then(json => {
```

```
                    console.log(json);
                    var redirectUriParsed = get("redirect_uri");
                    localStorage.setItem('token', json["data"].sessionId);
                    localStorage.setItem('csrfToken', json["data"].csrfToken);
                    window.location.href = redirectUriParsed + "?code=" +
json["data"].code;
                });
        } catch (error) {
            console.log("Ошибка", error);
            localStorage.clear();
        }
    }

    function escapeHtml(unsafe) {
        return unsafe
            .replace(/&/g, "&amp;")
            .replace(/</g, "&lt;")
            .replace(/>/g, "&gt;")
            .replace(/"/g, "&quot;")
            .replace(/'/g, "&#039;");
    }

    function get(name) {
        if (name = (new RegExp('[?&]' + encodeURIComponent(name) +
'=([^&]*)')).exec(location.search))
            return decodeURIComponent(name[1]);
    }
</script>
<style>
    .result-align {
        text-align: left;
    }

    .layer-left {
        text-align: center;
        width: 20%;
        margin: auto;
        margin-top: 50px;

    }

    #error-message {
        color: red;
    }

    .group-text-size {
        width: 110px;
    }
</style>
</html>
```

login.html

```
<!doctype html>
<html lang="en">
<script>
    const token = localStorage.getItem('token');
    if (token) {
        try {
            fetch('http://127.0.0.1:8443/session', {
```

```
                method: 'POST',
                headers: {
                    'Content-type': 'application/json; charset=UTF-8',
                    'Authorization': localStorage.getItem('token'),
                    'X-CSRF-TOKEN': localStorage.getItem("csrfToken")
                },
            })
                .then(response => response.json())
                .then(json => {
                    if (json.status === "OK") {
                        window.location.href = './index.html'
                    } else {
                        localStorage.clear();
                    }
                    console.log(json);
                });
        } catch (error) {
            console.log("Ошибка", e);
            localStorage.clear();
        }
    }
</script>
<head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-
fit=no">

    <!-- Bootstrap CSS -->
    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/css/bootstrap.min.css"
          integrity="sha384-
TX8t27EcRE3e/ihU7zmQxVncDAy5uIKz4rEkgIXeMed4M0jlfIDPvg6uqKI2xXr2" cros-
sorigin="anonymous">

    <title>Авторизация</title>
</head>
<body>
<nav class="navbar navbar-expand-lg navbar navbar navbar-dark bg-dark">
    <a class="navbar-brand" href="#">ЕЗСИАА</a>
</nav>
<div id="hidden-div-profile" class="layer-left">
    <h1>Авторизация</h1>
    <span id="error-message" style="display: none">Неверный логин или
пароль</span>
    <div class="input-group mb-3">
        <div class="input-group-prepend">
            <span class="input-group-text group-text-size" id="basic-
addon3">Логин</span>
        </div>
        <input id="loginSingIn" placeholder="Введите логин" class="form-control
result-align" aria-label="Username"
               aria-describedby="basic-addon1">
    </div>

    <div class="input-group mb-3">
        <div class="input-group-prepend">
            <span class="input-group-text group-text-size" id="basic-
addon3">Пароль</span>
        </div>
        <input type="password" id="passwordSingIn" placeholder="Введите пароль"
class="form-control result-align"
               aria-label="Username" aria-describedby="basic-addon1">
    </div>
```

```
    <button type="submit" onclick="f()" class="btn btn-dark">Войти</button>
    <a href="register.html">Регистрация</a>
</div>
<script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"
        integrity="sha384-
DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkfj"
        crossorigin="anonymous"></script>
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/js/bootstrap.bundle.min.js"
        integrity="sha384-
ho+j7jyWK8fNQe+A12Hb8AhRq26LrZ/JpcUGGOn+Y7RsweNrtN/tE3MoK7ZeZDyx"
        crossorigin="anonymous"></script>
</body>
<script>
    function escapeHtml(unsafe) {
        return unsafe
            .replace(/&/g, "&amp;")
            .replace(/</g, "&lt;")
            .replace(/>/g, "&gt;")
            .replace(/"/g, "&quot;")
            .replace(/'/g, "&#039;");
    }

    function f() {
        const loginSingIn = es-
capeHtml(document.getElementById("loginSingIn").value);
        const passwordSingIn = es-
capeHtml(document.getElementById("passwordSingIn").value);

        try {
            fetch('http://127.0.0.1:8443/login', {
                method: 'POST',
                body: JSON.stringify({
                    username: loginSingIn,
                    password: passwordSingIn
                }),
                headers: {
                    'Content-Type': 'application/json',
                    'Authorization': localStorage.getItem('token')
                },
            })
                .then(response => response.json())
                .then(json => {
                    try {
                        console.log(json);
                        if (json["status"] !== "OK") {
                            throw json["status"];
                        }
                        localStorage.setItem('token', json["data"].sessionId);
                        localStorage.setItem('csrfToken', json["data"].csrfToken);
                        console.log('Успех:', JSON.stringify(json));
                        window.location.href = './index.html'
                    } catch (error) {
                        console.error('Ошибка:', error);
                        var y = document.getElementById("error-message");
                        y.style.display = "block";
                    }
                });
        } catch (error) {
            console.error('Ошибка:', error);
            var y = document.getElementById("error-message").value;
            localStorage.clear();
        }
    }
```

```
</script>
<style>
    .result-align {
        text-align: left;
    }

    .layer-left {
        text-align: center;
        width: 20%;
        margin: auto;
        margin-top: 50px;

    }

    #error-message {
        color: red;
    }

    .group-text-size {
        width: 110px;
    }
</style>

</html>
```

register.html

```
<!doctype html>
<html lang="en">
<script>
    const token = localStorage.getItem('token');
    if (token) {
        try {
            fetch('http://127.0.0.1:8443/session', {
                method: 'POST',
                headers: {
                    'Content-type': 'application/json; charset=UTF-8',
                    'Authorization': localStorage.getItem('token'),
                    'X-CSRF-TOKEN': localStorage.getItem("csrfToken")
                },
            })
                .then(response => response.json())
                .then(json => {
                    if (json.status === "OK") {
                        window.location.href = './index.html'
                    } else {
                        localStorage.clear();
                    }
                    console.log(json);
                });
        } catch (error) {
            console.log("Ошибка", e);
            localStorage.clear();
        }
    }
</script>
<head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
```

```
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-
fit=no">

    <!-- Bootstrap CSS -->
    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/css/bootstrap.min.css"
          integrity="sha384-
TX8t27EcRE3e/ihU7zmQxVncDAy5uIKz4rEkgIXeMed4M0jlfIDPvg6uqKI2xXr2" cros-
sorigin="anonymous">

    <title>Регистрация</title>
</head>
<body>
<nav class="navbar navbar-expand-lg navbar navbar navbar-dark bg-dark">
    <a class="navbar-brand" href="./index.html">ЕЗСИАА</a>
</nav>
<div id="hidden-div-profile" class="layer-left" style="display: block">
    <h1>Регистрация</h1>
    <div class="input-group mb-3">
        <div class="input-group-prepend">
            <span class="input-group-text group-text-size" id="basic-
addon3">Логин</span>
        </div>
        <input id="login" placeholder="Введите логин" class="form-control result-
align" aria-label="Username"
               aria-describedby="basic-addon1">
    </div>

    <div class="input-group mb-3">
        <div class="input-group-prepend">
            <span class="input-group-text group-text-size" id="basic-
addon3">Пароль</span>
        </div>
        <input type="password" id="password" placeholder="Введите пароль"
class="form-control result-align"
               aria-label="Username" aria-describedby="basic-addon1">
    </div>

    <div class="input-group mb-3">
        <div class="input-group-prepend">
            <span class="input-group-text group-text-size" id="basic-
addon3">Пароль</span>
        </div>
        <input type="password" id="passwordRepeat" placeholder="Введите пароль"
class="form-control result-align"
               aria-label="Username" aria-describedby="basic-addon1">
    </div>
    <button type="submit" onclick="f()" class="btn btn-dark">Регистрация</button>
    <a href="login.html">Войти</a>
</div>
<div id="success-singup" class="layer-left" style="display: none">
    <p>Вы успешно зарегистрировались!</p>
    <a href="login.html">Войти</a>
</div>
<script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"
        integrity="sha384-
DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkfj"
        crossorigin="anonymous"></script>
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/js/bootstrap.bundle.min.js"
        integrity="sha384-
ho+j7jyWK8fNQe+A12Hb8AhRq26LrZ/JpcUGGOn+Y7RsweNrtN/tE3MoK7ZeZDyx"
        crossorigin="anonymous"></script>
</body>
```

```
<script>
    function escapeHtml(unsafe) {
        return unsafe
            .replace(/&/g, "&amp;")
            .replace(/</g, "&lt;")
            .replace(/>/g, "&gt;")
            .replace(/"/g, "&quot;")
            .replace(/'/g, "&#039;");
    }

    function f() {
        const url = 'http://127.0.0.1:8443/singup';

        const loginSingUp = escapeHtml(document.getElementById("login").value);
        const passwordSingUp = es-
capeHtml(document.getElementById("password").value);
        const passwordRepeatSingUp = es-
capeHtml(document.getElementById("passwordRepeat").value);

        try {
            const response = fetch(url, {
                method: 'POST',
                credentials: 'include',
                body: JSON.stringify({
                    name: loginSingUp,
                    password: passwordSingUp,
                    passwordRepeat: passwordRepeatSingUp
                }),
                headers: {
                    'Content-Type': 'application/json'
                }
            })
                .then(response => response.json())
                .then(json => {
                    console.log('Успех:', JSON.stringify(json));
                    var x = document.getElementById("success-singup");
                    x.style.display = "block";
                    var y = document.getElementById("hidden-div-profile");
                    y.style.display = "none";
                });
        } catch (error) {
            console.error('Ошибка:', error);
        }
    }
</script>
<style>
    .result-align {
        text-align: left;
    }

    .layer-left {
        text-align: center;
        width: 20%;
        margin: auto;
        margin-top: 50px;

    }

    #error-message {
        color: red;
    }

    #success-singup {
        color: blue;
```

```
    }

    .group-text-size {
        width: 110px;
    }
</style>
</html>
```

PasswordValidation.java

```
package com.web.esia.anntotation;

import javax.validation.Constraint;
import javax.validation.Payload;
import java.lang.annotation.*;

import static java.lang.annotation.ElementType.FIELD;
import static java.lang.annotation.RetentionPolicy.RUNTIME;

@Documented
@Target(FIELD)
@Retention(RUNTIME)
@Constraint(validatedBy = PasswordValidator.class)
public @interface PasswordValidation {
    public String value();

    String message() default "{com.web.esia.PasswordValidation.message}";

    Class<?>[] groups() default {};

    Class<? extends Payload>[] payload() default {};


}
```

PasswordValidator.java

```
package com.web.esia.anntotation;

import javax.validation.ConstraintValidator;
import javax.validation.ConstraintValidatorContext;

public class PasswordValidator implements ConstraintValidator<PasswordValidation,
String> {

    @Override
    public boolean isValid(String value, ConstraintValidatorContext context) {
        return value.length() >= 8 && value.length() <= 64;
    }
}
```

CorsConfiguration.java

```
package com.web.esia.conf;
```

```
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.servlet.config.annotation.CorsRegistry;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurerAdapter;

@Configuration
public class CorsConfiguration {

    @Bean
    public WebMvcConfigurer corsConfigurer() {
        return new WebMvcConfigurerAdapter() {
            @Override
            public void addCorsMappings(CorsRegistry registry) {
                registry.addMapping("/**")
                        .allowCredentials(true).maxAge(3600);
            }
        };
    }
}
```

AccessTokenResponse.java

```
package com.web.esia.controller.model;

import lombok.AllArgsConstructor;
import lombok.Getter;

import java.util.UUID;

@Getter
@AllArgsConstructor
public class AccessTokenResponse {
    private final UUID accessToken;
    private final String idToken;
    private final String tokenType;
}
```

AllClientsResponse.java

```
package com.web.esia.controller.model;

import lombok.AllArgsConstructor;
import lombok.Getter;

import java.util.List;

@Getter
@AllArgsConstructor
public class AllClientsResponse {
    private List<String> clientList;
}
```

ClientRequest.java

```
package com.web.esia.controller.model;

import lombok.Getter;
import lombok.Setter;

import javax.validation.constraints.NotEmpty;

@Setter
@Getter
public class ClientRequest {
    @NotEmpty
    private String name;
    @NotEmpty
    private String redirectUri;
}
```

ClientResponse.java

```
package com.web.esia.controller.model;

import lombok.AllArgsConstructor;
import lombok.Getter;

@Getter
@AllArgsConstructor
public class ClientResponse {
    private final String clientId;
    private final String clientSecret;
}
```

GenerateAccessTokenRequest.java

```
package com.web.esia.controller.model;

import lombok.Getter;
import lombok.Setter;

@Getter
@Setter
public class GenerateAccessTokenRequest {
    private String clientId;
    private String code;
    private String clientSecret;
}
```

LoginRequest.java

```
package com.web.esia.controller.model;
```

```
import lombok.Getter;
import lombok.Setter;

@Getter
@Setter
public class LoginRequest {
    private String username;
    private String password;
}
```

LoginResponse.java

```
package com.web.esia.controller.model;

import lombok.AllArgsConstructor;
import lombok.Getter;

@Getter
@AllArgsConstructor
public class LoginResponse {
    private String sessionId;
    private String csrfToken;
}
```

OAuthLoginResponse.java

```
package com.web.esia.controller.model;

import lombok.AllArgsConstructor;
import lombok.Getter;

@Getter
@AllArgsConstructor
public class OAuthLoginResponse {
    private final String code;
    private final String sessionId;
    private final String csrfToken;
}
```

OAuthSessionResponse.java

```
package com.web.esia.controller.model;

import lombok.AllArgsConstructor;
import lombok.Getter;

@Getter
@AllArgsConstructor
public class OAuthSessionResponse {
    private String code;
}
```

ResourceRequest.java

```
package com.web.esia.controller.model;

import lombok.AllArgsConstructor;
import lombok.Getter;

@Getter
@AllArgsConstructor
public class ResourceRequest {
    private final String accessToken;
}
```

ResourceResponse.java

```
package com.web.esia.controller.model;

import lombok.AllArgsConstructor;
import lombok.Getter;

@Getter
@AllArgsConstructor
public class ResourceResponse {
    private final String username;
    private final String city;
    private final String name;
    private final String hobby;
}
```

SessionRequest.java

```
package com.web.esia.controller.model;

import lombok.Getter;
import lombok.Setter;

@Setter
@Getter
public class SessionRequest {
    private String sessionId;
}
```

SingUpRequest.java

```
package com.web.esia.controller.model;

import lombok.Getter;
import lombok.Setter;

@Getter
@Setter
```

```
public class SingUpRequest {
    private String name;
    private String password;
    private String passwordRepeat;
}
```

StatusResponse.java

```
package com.web.esia.controller.model;

public enum StatusResponse {
    OK,
    BAD_REQUEST,
    GENERAL_ERROR,
    UNKNOWN_EXCEPTION
}
```

StatusSession.java

```
package com.web.esia.controller.model;

public enum StatusSession {
    ACTIVE,
    INVALIDATE
}
```

UserChangeRequest.java

```
package com.web.esia.controller.model;

import lombok.Getter;
import lombok.Setter;

@Getter
@Setter
public class UserChangeRequest {
    private String hobby;
    private String city;
    private String name;
}
```

UserInfoResponse.java

```
package com.web.esia.controller.model;

import lombok.AllArgsConstructor;
import lombok.Getter;

@Getter
@AllArgsConstructor
```

```
public class UserInfoResponse {
    private final String idToken;
}
```

AbstractController.java

```
package com.web.esia.controller;

import com.web.esia.model.ServerResponse;

import static com.web.esia.controller.model.StatusResponse.OK;

public abstract class AbstractController {
    public ServerResponse<Void> emptyOk() {
        return new ServerResponse<>(OK);
    }
}
```

AuthorizeController.java

```
package com.web.esia.controller;

import com.web.esia.controller.model.LoginRequest;
import com.web.esia.controller.model.LoginResponse;
import com.web.esia.controller.model.OAuthLoginResponse;
import com.web.esia.model.ServerResponse;
import com.web.esia.model.SessionProtection;
import com.web.esia.service.AuthorizeService;
import lombok.RequiredArgsConstructor;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

import javax.servlet.http.HttpServletResponse;

import static com.web.esia.controller.model.StatusResponse.OK;
import static org.springframework.web.util.HtmlUtils.htmlEscape;

@RestController
@RequiredArgsConstructor
public class AuthorizeController extends AbstractController {
    private final AuthorizeService authorizeService;

    @PostMapping(value = "/oauth/login")
    public ServerResponse<OAuthLoginResponse> authorize(@RequestParam(value =
"client_id") String clientId,
                                                        @RequestParam(value = "re-
direct_uri") String redirectUri,
                                                        @RequestBody LoginRequest
loginRequest,
                                                        HttpServletResponse re-
sponse) {
        clientId = htmlEscape(clientId);
        redirectUri = htmlEscape(redirectUri);
        loginRequest.setUsername(htmlEscape(loginRequest.getUsername()));
        loginRequest.setPassword(htmlEscape(loginRequest.getPassword()));
```

```
        return new ServerResponse<OAuthLoginResponse>(OK)
                .withData(authorizeService.oauthLogin(loginRequest, clientId, re-
sponse, redirectUri));
    }

    @PostMapping(value = "/login")
    public ServerResponse<LoginResponse> login(@RequestBody LoginRequest loginRe-
quest,
                                               HttpServletResponse response) {
        loginRequest.setUsername(htmlEscape(loginRequest.getUsername()));
        loginRequest.setPassword(htmlEscape(loginRequest.getPassword()));
        SessionProtection sessionProtection = authorizeService.login(loginRequest,
response);
        return new ServerResponse<LoginResponse>(OK)
                .withData(new LoginResponse(sessionProtection.getSessionId(), ses-
sionProtection.getCsrfToken().toString()));
    }
}
```

ClientController.java

```
package com.web.esia.controller;

import com.web.esia.controller.model.AllClientsResponse;
import com.web.esia.controller.model.ClientRequest;
import com.web.esia.controller.model.ClientResponse;
import com.web.esia.model.ServerResponse;
import com.web.esia.service.ClientService;
import com.web.esia.service.SessionAuthService;
import lombok.RequiredArgsConstructor;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestHeader;
import org.springframework.web.bind.annotation.RestController;

import static com.web.esia.controller.model.StatusResponse.OK;
import static org.springframework.web.util.HtmlUtils.htmlEscape;

@RestController
@RequiredArgsConstructor(onConstructor = @__(@Autowired))
public class ClientController {
    private final ClientService clientService;
    private final SessionAuthService sessionAuthService;

    @PostMapping(value = "/client/create")
    public ServerResponse<ClientResponse> createClient(@Validated @RequestBody
ClientRequest clientRequest) {
        clientRequest.setName(htmlEscape(clientRequest.getName()));
        clientRequest.setRedirectUri(htmlEscape(clientRequest.getRedirectUri()));
        return new ServerResponse<ClientResponse>(OK)
                .withData(clientService.createClient(clientRequest));
    }

    @PostMapping(value = "/client/fetch/all")
    public ServerResponse<AllClientsResponse> fetchAllClient-
Name(@RequestHeader("Authorization") String authorization,

@RequestHeader("X-CSRF-TOKEN") String csrfToken) {
```

```
        authorization = htmlEscape(authorization);
        csrfToken = htmlEscape(csrfToken);
        sessionAuthService.sessionAuthorize(authorization, csrfToken);
        return new ServerResponse<AllClientsResponse>(OK)
                .withData(clientService.fetchClientsName());
    }


}
```

LogoutController.java

```
package com.web.esia.controller;

import com.web.esia.model.ServerResponse;
import com.web.esia.model.Session;
import com.web.esia.service.SessionAuthService;
import com.web.esia.service.SessionService;
import lombok.RequiredArgsConstructor;
import lombok.extern.slf4j.Slf4j;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestHeader;
import org.springframework.web.bind.annotation.RestController;

import static com.web.esia.controller.model.StatusSession.INVALIDATE;
import static org.springframework.web.util.HtmlUtils.htmlEscape;

@Slf4j
@RestController
@RequiredArgsConstructor(onConstructor = @__(@Autowired))
public class LogoutController extends AbstractController {
    private final SessionAuthService sessionAuthService;
    private final SessionService sessionService;

    @GetMapping(value = "/logout")
    public ServerResponse<Void> logout(@RequestHeader("Authorization") String au-
thorization,
                                       @RequestHeader("X-CSRF-TOKEN") String
csrfToken) {
        authorization = htmlEscape(authorization);
        csrfToken = htmlEscape(csrfToken);
        sessionAuthService.sessionAuthorize(authorization, csrfToken);
        Session session = sessionService.fetchSession(authorization);
        sessionService.setStatus(INVALIDATE, authorization);
        log.info("Пользователь '{}' завершил сессию '{}'", session.getUsername(),
authorization);
        return emptyOk();
    }
}
```

ModuleExceptionHandler.java

```
package com.web.esia.controller;

import com.web.esia.exception.ModuleException;
import com.web.esia.model.ServerResponse;
```

```
import lombok.extern.slf4j.Slf4j;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.http.converter.HttpMessageNotReadableException;
import org.springframework.web.HttpMediaTypeNotSupportedException;
import org.springframework.web.bind.MethodArgumentNotValidException;
import org.springframework.web.bind.MissingServletRequestParameterException;
import org.springframework.web.bind.ServletRequestBindingException;
import org.springframework.web.bind.annotation.ControllerAdvice;
import org.springframework.web.bind.annotation.ExceptionHandler;
import
org.springframework.web.method.annotation.MethodArgumentTypeMismatchException;

import javax.validation.ConstraintViolationException;

import static com.web.esia.controller.model.StatusResponse.BAD_REQUEST;
import static com.web.esia.controller.model.StatusResponse.UNKNOWN_EXCEPTION;

@Slf4j
@ControllerAdvice
public class ModuleExceptionHandler {

    @ExceptionHandler(ModuleException.class)
    protected ResponseEntity<ServerResponse<Void>> handleModuleExcep-
tion(ModuleException ex) {
        return new ResponseEntity<>(
                new ServerResponse<>(ex.getStatus()),
                HttpStatus.OK
        );
    }

    @ExceptionHandler({
            ConstraintViolationException.class,
            HttpMediaTypeNotSupportedException.class,
            HttpMessageNotReadableException.class,
            MethodArgumentNotValidException.class,
            MissingServletRequestParameterException.class,
            ServletRequestBindingException.class,
            MethodArgumentTypeMismatchException.class
    })
    protected ResponseEntity<ServerResponse<Void>> handleBadRequestExcep-
tion(Exception ex) {
        log.warn("Ошибка при разборе параметров запроса", ex);
        return new ResponseEntity<>(
                new ServerResponse<>(BAD_REQUEST),
                HttpStatus.OK
        );
    }

    @ExceptionHandler(Throwable.class)
    protected ResponseEntity<ServerResponse<Void>> handleUnknownRequest(Throwable
ex) {
        log.error("Произошла неизвестна ошибка", ex);
        return new ResponseEntity<>(
                new ServerResponse<>(UNKNOWN_EXCEPTION),
                HttpStatus.OK
        );
    }

}
```

ResourceController.java

```
package com.web.esia.controller;

import com.web.esia.controller.model.ResourceResponse;
import com.web.esia.model.ServerResponse;
import com.web.esia.service.ResourceService;
import lombok.RequiredArgsConstructor;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestHeader;
import org.springframework.web.bind.annotation.RestController;

import static com.web.esia.controller.model.StatusResponse.OK;
import static org.springframework.web.util.HtmlUtils.htmlEscape;

@RestController
@RequiredArgsConstructor(onConstructor = @__(@Autowired))
public class ResourceController {
    private final ResourceService resourceService;

    @PostMapping(value = "/user/resources")
    public ServerResponse<ResourceResponse>
getResource(@RequestHeader("Authorization") String authorization,
                                                    @RequestHeader("X-CSRF-
TOKEN") String csrfToken) {
        authorization = htmlEscape(authorization);
        csrfToken = htmlEscape(csrfToken);
        return new ServerResponse<ResourceResponse>(OK)
                .withData(resourceService.getResource(authorization, csrfToken));
    }

    @PostMapping(value = "/user/resources/by/access/token")
    public ServerResponse<ResourceResponse> getResourceByAc-
cessToken(@RequestHeader("Authorization") String authorization) {
        authorization = htmlEscape(authorization);
        return new ServerResponse<ResourceResponse>(OK)

.withData(resourceService.getResourceByAccessToken(authorization));
    }
}
```

SessionController.java

```
package com.web.esia.controller;

import com.web.esia.controller.model.OAuthSessionResponse;
import com.web.esia.model.ServerResponse;
import com.web.esia.model.Session;
import com.web.esia.service.AuthorizeService;
import com.web.esia.service.SessionService;
import lombok.RequiredArgsConstructor;
import lombok.extern.slf4j.Slf4j;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestHeader;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;
```

```
import static com.web.esia.controller.model.StatusResponse.OK;
import static org.springframework.web.util.HtmlUtils.htmlEscape;

@Slf4j
@RestController
@RequiredArgsConstructor(onConstructor = @__(@Autowired))
public class SessionController extends AbstractController {
    private final SessionService sessionService;
    private final AuthorizeService authorizeService;

    @PostMapping(value = "session")
    public ServerResponse<Void> check(@RequestHeader(value = "Authorization")
String sessionId) {
        sessionId = htmlEscape(sessionId);
        Session session = sessionService.fetchSession(sessionId);
        log.info("Пользователь '{}' авторизован по сессии '{}'", ses-
sion.getUsername(), session.getSessionId());
        return emptyOk();
    }

    @PostMapping(value = "session/oauth")
    public ServerResponse<OAuthSessionResponse> checkOAuth(@RequestParam(value =
"client_id") String clientId,
                                                           @RequestParam(value =
"redirect_uri") String redirectUri,
                                                           @RequestHeader(value =
"Authorization") String sessionId,
                                                           @RequestHeader(value =
"X-CSRF-TOKEN") String csrfToken) {
        clientId = htmlEscape(clientId);
        redirectUri = htmlEscape(redirectUri);
        sessionId = htmlEscape(sessionId);
        return new ServerResponse<OAuthSessionResponse>(OK)
                .withData(new OAuthSessionRe-
sponse(authorizeService.oauthAuthorizeBySession(clientId, redirectUri, sessionId,
csrfToken)));
    }
}
```

SingUpController.java

```
package com.web.esia.controller;

import com.web.esia.controller.model.SingUpRequest;
import com.web.esia.model.ServerResponse;
import com.web.esia.service.SingUpService;
import lombok.RequiredArgsConstructor;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RestController;

import static org.springframework.web.util.HtmlUtils.htmlEscape;

@RestController
@RequiredArgsConstructor(onConstructor = @__(@Autowired))
public class SingUpController extends AbstractController {
    private final SingUpService singUpService;
```

```
    @PostMapping(value = "/singup")
    public ServerResponse<Void> singUp(@RequestBody SingUpRequest request) {
        request.setName(htmlEscape(request.getName()));
        request.setPassword(htmlEscape(request.getPassword()));
        request.setPasswordRepeat(htmlEscape(request.getPasswordRepeat()));
        singUpService.singUp(request);
        return emptyOk();
    }
}
```

TokenController.java

```
package com.web.esia.controller;

import com.web.esia.controller.model.GenerateAccessTokenRequest;
import com.web.esia.controller.model.AccessTokenResponse;
import com.web.esia.model.ServerResponse;
import com.web.esia.service.GenerateAccessTokenService;
import lombok.RequiredArgsConstructor;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import static com.web.esia.controller.model.StatusResponse.OK;
import static org.springframework.web.util.HtmlUtils.htmlEscape;

@RestController
@RequiredArgsConstructor(onConstructor = @__(@Autowired))
public class TokenController {
    private final GenerateAccessTokenService generateAccessTokenService;

    @PostMapping("/token")
    public ServerResponse<AccessTokenResponse> generateAccessToken(
            @RequestBody GenerateAccessTokenRequest generateAccessTokenRequest) {
        generateAccessToken-
Request.setClientId(htmlEscape(generateAccessTokenRequest.getClientId()));
        generateAccessToken-
Request.setClientSecret(htmlEscape(generateAccessTokenRequest.getClientSecret()));
        generateAccessToken-
Request.setCode(htmlEscape(generateAccessTokenRequest.getCode()));
        return new ServerResponse<AccessTokenResponse>(OK)

.withData(generateAccessTokenService.generateAccessToken(generateAccessTokenReques
t));
    }
}
```

UserController.java

```
package com.web.esia.controller;

import com.web.esia.controller.model.UserChangeRequest;
import com.web.esia.model.ServerResponse;
import com.web.esia.service.ResourceService;
import com.web.esia.service.SessionAuthService;
import lombok.RequiredArgsConstructor;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.PostMapping;
```

```
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestHeader;
import org.springframework.web.bind.annotation.RestController;

import static org.springframework.web.util.HtmlUtils.htmlEscape;

@RestController
@RequiredArgsConstructor(onConstructor = @__(@Autowired))
public class UserController extends AbstractController {
    private final ResourceService resourceService;
    private final SessionAuthService sessionAuthService;

    @PostMapping(value = "/user/change")
    public ServerResponse<Void> changeUserData(@RequestBody UserChangeRequest us-
erChangeRequest,
                                               @RequestHeader("Authorization")
String authorization,
                                               @RequestHeader("X-CSRF-TOKEN")
String csrfToken) {
        authorization = htmlEscape(authorization);
        csrfToken = htmlEscape(csrfToken);
        userChangeRequest.setCity(htmlEscape(userChangeRequest.getCity()));
        userChangeRequest.setHobby(htmlEscape(userChangeRequest.getHobby()));
        userChangeRequest.setName(htmlEscape(userChangeRequest.getName()));
        sessionAuthService.sessionAuthorize(authorization, csrfToken);
        resourceService.updateUserData(userChangeRequest, authorization);
        return emptyOk();
    }
}
```

UserInfoController.java

```
package com.web.esia.controller;

import com.web.esia.controller.model.UserInfoResponse;
import com.web.esia.model.ServerResponse;
import com.web.esia.service.IdTokenService;
import lombok.RequiredArgsConstructor;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestHeader;
import org.springframework.web.bind.annotation.RestController;

import static com.web.esia.controller.model.StatusResponse.OK;

@RestController
@RequiredArgsConstructor(onConstructor = @__(@Autowired))
public class UserInfoController {
    private final IdTokenService idTokenService;

    @PostMapping(value = "/userinfo")
    public ServerResponse<UserInfoResponse> fetchUserIn-
fo(@RequestHeader("Authorization") String authorization) {
        return new ServerResponse<UserInfoResponse>(OK)
                .withData(idTokenService.fetchUserInfo(authorization));
    }
}
```

ClientDao.java

```
package com.web.esia.dao;

import com.web.esia.model.Client;
import lombok.RequiredArgsConstructor;
import lombok.extern.slf4j.Slf4j;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.jdbc.core.RowMapper;
import org.springframework.jdbc.core.namedparam.MapSqlParameterSource;
import org.springframework.jdbc.core.namedparam.NamedParameterJdbcTemplate;
import org.springframework.stereotype.Repository;

import javax.swing.text.html.Option;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.List;
import java.util.Map;
import java.util.Optional;

@Slf4j
@Repository
@RequiredArgsConstructor(onConstructor = @__(@Autowired))
public class ClientDao {
    private final NamedParameterJdbcTemplate namedParameterJdbcTemplate;
    private final ClientsMapper clientsMapper = new ClientsMapper();

    public boolean isExistClientIdWithClientSecret(String clientId, String cli-
entSecret) {
        try {
            String sql = "SELECT * FROM CLIENTS WHERE CLIENT_ID=:clientId AND CLI-
ENT_SECRET=:clientSecret";

            MapSqlParameterSource mapSqlParameterSource = new MapSqlParameter-
Source();
            mapSqlParameterSource
                    .addValue("clientId", clientId)
                    .addValue("clientSecret", clientSecret);

            namedParameterJdbcTemplate.queryForObject(sql, mapSqlParameterSource,
clientsMapper);
            return true;
        } catch (Exception ex) {
            return false;
        }
    }

    public Optional<Client> fetchClient(String clientId) {
        try {
            String sql = "SELECT * FROM CLIENTS WHERE CLIENT_ID=:clientId";

            MapSqlParameterSource mapSqlParameterSource = new MapSqlParameter-
Source();
            mapSqlParameterSource
                    .addValue("clientId", clientId);

            return Option-
al.ofNullable(namedParameterJdbcTemplate.queryForObject(sql, mapSqlParameter-
Source, clientsMapper));
        } catch (Exception ex) {
            log.error("Клиент с clientId '{} не найден", clientId);
            return Optional.empty();
```

```
        }
    }

    public void save(String clientId, String clientSecret, String name, String re-
directUrl) {
        String sql = "INSERT INTO CLIENTS(CLIENT_ID, CLIENT_SECRET, CLIENT_NAME,
CLIENT_REDIRECT_URL) VALUES(:clientId, :clientSecret, :name, :redirectUrl)";

        MapSqlParameterSource mapSqlParameterSource = new MapSqlParameterSource()
                .addValue("clientId", clientId)
                .addValue("name", name)
                .addValue("redirectUrl", redirectUrl)
                .addValue("clientSecret", clientSecret);

        namedParameterJdbcTemplate.update(sql, mapSqlParameterSource);
        log.info("Создан новый клиент '{}'", name);
    }

    public List<String> fetchClientsName() {
        String sql = "SELECT CLIENT_NAME FROM CLIENTS";

        return namedParameterJdbcTemplate.query(sql,
                (rs, rowNum) -> rs.getString(1));
    }

    static class ClientsMapper implements RowMapper<Client> {

        @Override
        public Client mapRow(ResultSet rs, int rowNum) throws SQLException {
            Client client = new Client();
            client.setClientId(rs.getString("CLIENT_ID"));
            client.setClientSecret(rs.getString("CLIENT_SECRET"));
            client.setName(rs.getString("CLIENT_NAME"));
            client.setRedirectUri(rs.getString("CLIENT_REDIRECT_URL"));
            return client;
        }
    }
}
```

CodeDao.java

```
package com.web.esia.dao;

import com.web.esia.exception.ModuleException;
import com.web.esia.model.Code;
import com.web.esia.model.StatusCode;
import lombok.RequiredArgsConstructor;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.jdbc.core.RowMapper;
import org.springframework.jdbc.core.namedparam.MapSqlParameterSource;
import org.springframework.jdbc.core.namedparam.NamedParameterJdbcTemplate;
import org.springframework.stereotype.Repository;

import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Timestamp;
import java.util.Date;
import java.util.UUID;

import static com.web.esia.controller.model.StatusResponse.BAD_REQUEST;
```

```
@Repository
@RequiredArgsConstructor(onConstructor = @__(@Autowired))
public class CodeDao {
    private final NamedParameterJdbcTemplate namedParameterJdbcTemplate;
    private final CodeRowMapper codeRowMapper = new CodeRowMapper();

    public Code getCode(String clientId, String code) {
        try {

            String sql = "SELECT * FROM CODE WHERE CODE=:code AND CLI-
ENT_ID=:clientId";

            MapSqlParameterSource mapSqlParameterSource = new MapSqlParameter-
Source()
                    .addValue("code", code)
                    .addValue("clientId", clientId);

            return namedParameterJdbcTemplate.queryForObject(sql, mapSqlParameter-
Source, codeRowMapper);
        } catch (Exception ex) {
            throw new ModuleException(BAD_REQUEST);
        }
    }

    public boolean isExistUnusedAndNotExpirationCodeFor(String username, String
clientId) {
        try {
            String sql = "SELECT * FROM CODE WHERE USERNAME=:username AND CLI-
ENT_ID=:clientId AND STATUS='NEW' AND EXPIRATION <= :date";

            MapSqlParameterSource mapSqlParameterSource = new MapSqlParameter-
Source()
                    .addValue("clinetId", clientId)
                    .addValue("username", username)
                    .addValue("date", new Timestamp(new Date().getTime()));

            namedParameterJdbcTemplate.queryForObject(sql, mapSqlParameterSource,
codeRowMapper);
            return true;
        } catch (Exception ex) {
            return false;
        }

    }

    public void save(UUID code, String username, String clientId) {
        String sql = "INSERT INTO CODE(CODE, USERNAME, EXPIRATION, CLIENT_ID, STA-
TUS) VALUES(:code, :username, :date, :clientId, 'NEW')";

        MapSqlParameterSource mapSqlParameterSource = new MapSqlParameterSource()
                .addValue("code", code.toString())
                .addValue("username", username)
                .addValue("clientId", clientId)
                .addValue("date", new Timestamp(new Date().getTime() + 2000));

        namedParameterJdbcTemplate.update(sql, mapSqlParameterSource);
    }

    public void setStatus(StatusCode status, String code) {
        String sql = "UPDATE CODE SET STATUS=:status WHERE CODE=:code";

        MapSqlParameterSource mapSqlParameterSource = new MapSqlParameterSource()
                .addValue("status", status.toString())
```

```
                .addValue("code", code);

        namedParameterJdbcTemplate.update(sql, mapSqlParameterSource);
    }

    static class CodeRowMapper implements RowMapper<Code> {

        @Override
        public Code mapRow(ResultSet rs, int rowNum) throws SQLException {
            Code code = new Code();
            code.setClientId(rs.getString("CLIENT_ID"));
            code.setCode(UUID.fromString(rs.getString("CODE")));
            code.setExpiration(rs.getTimestamp("EXPIRATION"));
            code.setStatusCode(StatusCode.valueOf(rs.getString("STATUS")));
            code.setUsername(rs.getString("USERNAME"));
            return code;
        }
    }
}
```

GenerateAccessTokenDao.java

```
package com.web.esia.dao;

import com.web.esia.exception.ModuleException;
import com.web.esia.model.AccessToken;
import lombok.RequiredArgsConstructor;
import lombok.extern.slf4j.Slf4j;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.jdbc.core.RowMapper;
import org.springframework.jdbc.core.namedparam.MapSqlParameterSource;
import org.springframework.jdbc.core.namedparam.NamedParameterJdbcTemplate;
import org.springframework.stereotype.Repository;

import javax.swing.text.html.Option;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.Optional;
import java.util.UUID;

import static com.web.esia.controller.model.StatusResponse.BAD_REQUEST;

@Slf4j
@Repository
@RequiredArgsConstructor(onConstructor = @__(@Autowired))
public class GenerateAccessTokenDao {
    private final NamedParameterJdbcTemplate namedParameterJdbcTemplate;
    private final AccessTokenRowMapper accessTokenRowMapper = new AccessTokenRow-
Mapper();

    public void save(UUID accessToken, String username) {
        try {
            String sql = "INSERT INTO ACCESS_TOKEN(ACCESS_TOKEN, USERNAME) VAL-
UES(:accessToken, :username)";

            MapSqlParameterSource mapSqlParameterSource = new MapSqlParameter-
Source()
                    .addValue("accessToken", accessToken.toString())
                    .addValue("username", username);
```

```
            namedParameterJdbcTemplate.update(sql, mapSqlParameterSource);
        } catch (Exception ex) {
            throw new ModuleException(BAD_REQUEST);
        }
    }

    public Optional<AccessToken> fetchAccessToken(String accessToken) {
        try {
            String sql = "SELECT * FROM ACCESS_TOKEN WHERE AC-
CESS_TOKEN=:accessToken";

            MapSqlParameterSource mapSqlParameterSource = new MapSqlParameter-
Source()
                    .addValue("accessToken", accessToken);

            return Option-
al.ofNullable(namedParameterJdbcTemplate.queryForObject(sql, mapSqlParameter-
Source, accessTokenRowMapper));
        } catch (Exception ex) {
            return Optional.empty();
        }
    }


    static class AccessTokenRowMapper implements RowMapper<AccessToken> {

        @Override
        public AccessToken mapRow(ResultSet rs, int rowNum) throws SQLException {
            AccessToken accessToken = new AccessToken();
            accessToken.setUsername(rs.getString("USERNAME"));
            ac-
cessToken.setAccessToken(UUID.fromString(rs.getString("ACCESS_TOKEN")));
            return accessToken;
        }
    }
}
```

IdTokenDao.java

```
package com.web.esia.dao;

import com.web.esia.model.Code;
import com.web.esia.model.IdToken;
import com.web.esia.model.Resource;
import com.web.esia.model.StatusCode;
import lombok.RequiredArgsConstructor;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.jdbc.core.RowMapper;
import org.springframework.jdbc.core.namedparam.MapSqlParameterSource;
import org.springframework.jdbc.core.namedparam.NamedParameterJdbcTemplate;
import org.springframework.stereotype.Repository;

import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.Optional;
import java.util.UUID;

@Repository
@RequiredArgsConstructor(onConstructor = @__(@Autowired))
public class IdTokenDao {
```

```
    private final NamedParameterJdbcTemplate namedParameterJdbcTemplate;
    private final IdTokenMapper idTokenMapper = new IdTokenMapper();

    public void save(String username, String idToken) {
        String sql = "INSERT INTO ID_TOKEN(USERNAME, ID_TOKEN) VALUES(:username,
:idToken)";

        MapSqlParameterSource mapSqlParameterSource = new MapSqlParameterSource()
                .addValue("username", username)
                .addValue("idToken", idToken);

        namedParameterJdbcTemplate.update(sql, mapSqlParameterSource);
    }

    public Optional<IdToken> fetchIdToken(String username) {
        try {
            String sql = "SELECT * FROM ID_TOKEN WHERE USERNAME=:username";

            MapSqlParameterSource mapSqlParameterSource = new MapSqlParameter-
Source()
                    .addValue("username", username);

            return Option-
al.ofNullable(namedParameterJdbcTemplate.queryForObject(sql, mapSqlParameter-
Source, idTokenMapper));
        } catch (Exception ex) {
            return Optional.empty();
        }
    }

    static class IdTokenMapper implements RowMapper<IdToken> {
        @Override
        public IdToken mapRow(ResultSet rs, int rowNum) throws SQLException {
            IdToken idToken = new IdToken();
            idToken.setUsername(rs.getString("USERNAME"));
            idToken.setIdToken(rs.getString("ID_TOKEN"));
            return idToken;
        }
    }
}
```

ResourceDao.java

```
package com.web.esia.dao;

import com.web.esia.controller.model.UserChangeRequest;
import com.web.esia.model.Resource;
import lombok.RequiredArgsConstructor;
import lombok.extern.slf4j.Slf4j;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.jdbc.core.RowMapper;
import org.springframework.jdbc.core.namedparam.MapSqlParameterSource;
import org.springframework.jdbc.core.namedparam.NamedParameterJdbcTemplate;
import org.springframework.stereotype.Repository;

import java.sql.ResultSet;
import java.sql.SQLException;

@Slf4j
@Repository
```

```
@RequiredArgsConstructor(onConstructor = @__(@Autowired))
public class ResourceDao {
    private final NamedParameterJdbcTemplate namedParameterJdbcTemplate;
    private final ResourceRowMapper resourceRowMapper = new ResourceRowMapper();

    public Resource fetchResource(String username) {
        try {
            String sql = "SELECT * FROM RESOURCE WHERE USERNAME=:username";
            MapSqlParameterSource mapSqlParameterSource = new MapSqlParameter-
Source()
                    .addValue("username", username);

            return namedParameterJdbcTemplate.queryForObject(sql, mapSqlParameter-
Source, resourceRowMapper);
        } catch (Exception ex) {
            log.error("Ошибка", ex);
            throw ex;
        }
    }

    public void save(String username, UserChangeRequest userChangeRequest) {
        String sql = "INSERT INTO RESOURCE(NAME, CITY, HOBBY, USERNAME) VAL-
UES(:name, :city, :hobby, :username)";

        MapSqlParameterSource mapSqlParameterSource = new MapSqlParameterSource()
                .addValue("name", userChangeRequest.getName())
                .addValue("username", username)
                .addValue("city", userChangeRequest.getCity())
                .addValue("hobby", userChangeRequest.getHobby());

        namedParameterJdbcTemplate.update(sql, mapSqlParameterSource);
    }

    public void update(String username, UserChangeRequest userChangeRequest) {
        String sql = "UPDATE RESOURCE SET NAME=:name, CITY=:city, HOBBY=:hobby
WHERE USERNAME=:username";

        MapSqlParameterSource mapSqlParameterSource = new MapSqlParameterSource()
                .addValue("name", userChangeRequest.getName())
                .addValue("username", username)
                .addValue("city", userChangeRequest.getCity())
                .addValue("hobby", userChangeRequest.getHobby());

        namedParameterJdbcTemplate.update(sql, mapSqlParameterSource);
    }

    static class ResourceRowMapper implements RowMapper<Resource> {

        @Override
        public Resource mapRow(ResultSet rs, int rowNum) throws SQLException {
            Resource resource = new Resource();
            resource.setName(rs.getString("NAME"));
            resource.setHobby(rs.getString("HOBBY"));
            resource.setCity(rs.getString("CITY"));
            resource.setUsername(rs.getString("USERNAME"));
            return resource;
        }
    }
}
```

SessionDao.java

```
package com.web.esia.dao;

import com.web.esia.controller.model.StatusSession;
import com.web.esia.model.Session;
import lombok.RequiredArgsConstructor;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.jdbc.core.RowMapper;
import org.springframework.jdbc.core.namedparam.MapSqlParameterSource;
import org.springframework.jdbc.core.namedparam.NamedParameterJdbcTemplate;
import org.springframework.stereotype.Repository;

import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.Optional;
import java.util.UUID;

@Repository
@RequiredArgsConstructor(onConstructor = @__(@Autowired))
public class SessionDao {
    private final NamedParameterJdbcTemplate namedParameterJdbcTemplate;
    private final SessionRowMapper sessionRowMapper = new SessionRowMapper();

    public void save(String sessionId, String username, UUID csrfToken) {
        String sql = "INSERT INTO SESSION(USERNAME, SESSION_ID, STATUS,
CSRF_TOKEN) VALUES(:username, :sessionId, :status, :csrfToken)";

        MapSqlParameterSource mapSqlParameterSource = new MapSqlParameterSource()
                .addValue("sessionId", sessionId)
                .addValue("username", username)
                .addValue("status", StatusSession.ACTIVE.toString())
                .addValue("csrfToken", csrfToken.toString());

        namedParameterJdbcTemplate.update(sql, mapSqlParameterSource);
    }

    public void setStatus(String sessionId, StatusSession statusSession) {
        String sql = "UPDATE SESSION SET STATUS=:status WHERE SES-
SION_ID=:sessionId";

        MapSqlParameterSource mapSqlParameterSource = new MapSqlParameterSource()
                .addValue("sessionId", sessionId)
                .addValue("status", statusSession.toString());

        namedParameterJdbcTemplate.update(sql, mapSqlParameterSource);
    }

    public Optional<Session> fetchSessionByUsername(String username) {
        try {
            String sql = "SELECT * FROM SESSION WHERE USERNAME=:username AND STA-
TUS='ACTIVE'";

            MapSqlParameterSource mapSqlParameterSource = new MapSqlParameter-
Source()
                    .addValue("username", username);

            Session session = namedParameterJdbcTemplate.queryForObject(sql, map-
SqlParameterSource, sessionRowMapper);
            return Optional.of(session);
        } catch (Exception ex) {
            return Optional.empty();
        }
    }
```

```
    public boolean isSessionExist(String sessionId) {
        try {
            String sql = "SELECT * FROM SESSION WHERE SESSION_ID=:sessionId";

            MapSqlParameterSource mapSqlParameterSource = new MapSqlParameter-
Source()
                    .addValue("sessionId", sessionId);

            namedParameterJdbcTemplate.queryForObject(sql, mapSqlParameterSource,
sessionRowMapper);

            return true;
        } catch (Exception ex) {
            return false;
        }
    }

    public Optional<Session> fetchSession(String sessionId) {
        try {
            String sql = "SELECT * FROM SESSION WHERE SESSION_ID=:sessionId";

            MapSqlParameterSource mapSqlParameterSource = new MapSqlParameter-
Source()
                    .addValue("sessionId", sessionId);

            Session session = namedParameterJdbcTemplate.queryForObject(sql, map-
SqlParameterSource, sessionRowMapper);
            return Optional.of(session);
        } catch (Exception ex) {
            return Optional.empty();
        }
    }

    static class SessionRowMapper implements RowMapper<Session> {

        @Override
        public Session mapRow(ResultSet rs, int rowNum) throws SQLException {
            Session session = new Session();
            session.setSessionId(rs.getString("SESSION_ID"));
            session.setUsername(rs.getString("USERNAME"));
            session.setStatus(StatusSession.valueOf(rs.getString("STATUS")));
            session.setCsrfToken(rs.getString("CSRF_TOKEN"));
            return session;
        }
    }
}
```

UserDao.java

```
package com.web.esia.dao;

import com.web.esia.model.StatusConfirmEmail;
import com.web.esia.model.User;
import lombok.RequiredArgsConstructor;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.dao.EmptyResultDataAccessException;
import org.springframework.jdbc.core.RowMapper;
import org.springframework.jdbc.core.namedparam.MapSqlParameterSource;
import org.springframework.jdbc.core.namedparam.NamedParameterJdbcTemplate;
```

```
import org.springframework.stereotype.Repository;

import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.Optional;

@Repository
@RequiredArgsConstructor(onConstructor = @__(@Autowired))
public class UserDao {
    private final NamedParameterJdbcTemplate namedParameterJdbcTemplate;
    private final UserRowMapper userRowMapper = new UserRowMapper();

    public void save(String username, String password, StatusConfirmEmail sta-
tusConfirmEmail) {
        String sql = "INSERT INTO USERS (USERNAME, PASSWORD, CONFIRM_EMAIL) " +
                "VALUES(:username, :secret, :statusConfirmEmail)";

        MapSqlParameterSource mapSqlParameterSource = new MapSqlParameterSource()
                .addValue("username", username)
                .addValue("secret", password)
                .addValue("statusConfirmEmail", statusConfirmEmail.name());

        namedParameterJdbcTemplate.update(sql, mapSqlParameterSource);
    }

    public boolean isUserExist(String username) {
        try {
            String sql = "SELECT * FROM USERS WHERE username=:username";

            MapSqlParameterSource mapSqlParameterSource = new MapSqlParameter-
Source()
                    .addValue("username", username);

            namedParameterJdbcTemplate.queryForObject(sql, mapSqlParameterSource,
userRowMapper);
            return true;
        } catch (EmptyResultDataAccessException ex) {
            return false;
        }
    }

    public Optional<User> fetchUser(String username) {
        try {

            String sql = "SELECT * FROM USERS WHERE USERNAME = :username";

            MapSqlParameterSource mapSqlParameterSource = new MapSqlParameter-
Source()
                    .addValue("username", username);

            User user = namedParameterJdbcTemplate.queryForObject(sql, mapSqlPa-
rameterSource, userRowMapper);
            return Optional.of(user);
        } catch (EmptyResultDataAccessException ex) {
            return Optional.empty();
        }
    }

    static class UserRowMapper implements RowMapper<User> {

        @Override
        public User mapRow(ResultSet rs, int rowNum) throws SQLException {
            User user = new User();
            user.setUsername(rs.getString("USERNAME"));
```

```
            user.setPassword(rs.getString("PASSWORD"));
            user.setEmail(rs.getString("EMAIL"));
            us-
er.setStatusConfirmEmail(StatusConfirmEmail.valueOf(rs.getString("CONFIRM_EMAIL"))
);
            return user;
        }
    }
}
```

ModuleException.java

```
package com.web.esia.exception;

import lombok.Getter;

@Getter
public class ModuleException extends RuntimeException {
    private final Enum<?> status;

    public <T extends Enum<?>> ModuleException(T status) {
        super(status.name());
        this.status = status;
    }

}
```

WebSecurityConfig.java

```
package com.web.esia.infra.security;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

@Configuration
public class WebSecurityConfig {
    @Bean
    public BCryptPasswordEncoder bCryptPasswordEncoder() {
        return new BCryptPasswordEncoder(10);
    }
}
```

AccessToken.java

```
package com.web.esia.model;

import lombok.Getter;
import lombok.Setter;

import java.util.UUID;

@Setter
@Getter
```

```
public class AccessToken {
    private String username;
    private UUID accessToken;
}
```

Client.java

```
package com.web.esia.model;

import lombok.Getter;
import lombok.Setter;

@Setter
@Getter
public class Client {
    private String clientId;
    private String clientSecret;
    private String name;
    private String redirectUri;
}
```

Code.java

```
package com.web.esia.model;

import lombok.Getter;
import lombok.Setter;

import java.util.Date;
import java.util.UUID;

@Setter
@Getter
public class Code {
    private String clientId;
    private Date expiration;
    private UUID code;
    private String username;
    private StatusCode statusCode;
}
```

IdToken.java

```
package com.web.esia.model;

import lombok.Getter;
import lombok.Setter;
import org.springframework.stereotype.Repository;

@Setter
@Getter
public class IdToken {
    private String username;
    private String idToken;
```

```
}
```

Resource.java

```
package com.web.esia.model;

import lombok.Getter;
import lombok.Setter;

@Getter
@Setter
public class Resource {
    private String username;
    private String name;
    private String hobby;
    private String city;
}
```

ServerResponse.java

```
package com.web.esia.model;

import com.fasterxml.jackson.annotation.JsonCreator;
import com.fasterxml.jackson.annotation.JsonPropertyOrder;
import com.fasterxml.jackson.databind.annotation.JsonSerialize;
import lombok.Getter;
import lombok.ToString;
import lombok.experimental.FieldNameConstants;
import org.springframework.web.bind.annotation.GetMapping;

import java.beans.ConstructorProperties;

@Getter
@ToString
@FieldNameConstants
@JsonPropertyOrder({"status", "data"})
public class ServerResponse<D> {
    private String status;
    private D data;

    public ServerResponse(Enum<?> status) {
        this(status.name());
    }

    public ServerResponse(Enum<?> status, D data) {
        this(status.name());
        this.data = data;
    }

    @JsonCreator
    @ConstructorProperties({"status"})
    public ServerResponse(String status) {
        this.status = status;
    }

    public ServerResponse<D> withData(D data) {
        this.data = data;
```

```
        return this;
    }
}
```

Session.java

```
package com.web.esia.model;

import com.web.esia.controller.model.StatusSession;
import lombok.Getter;
import lombok.Setter;

@Setter
@Getter
public class Session {
    private String username;
    private String sessionId;
    private StatusSession status;
    private String csrfToken;
}
```

SessionProtection.java

```
package com.web.esia.model;

import lombok.AllArgsConstructor;
import lombok.Getter;

import java.util.UUID;

@Getter
@AllArgsConstructor
public class SessionProtection {
    private final String sessionId;
    private final UUID csrfToken;
}
```

StatusCode.java

```
package com.web.esia.model;

public enum StatusCode {
    NEW,
    USED,
    EXPIRATION
}
```

StatusConfirmEmail.java

```
package com.web.esia.model;
```

```
public enum StatusConfirmEmail {
    NEW,
    SEND_EMAIL,
    CONFIRMED
}
```

User.java

```
package com.web.esia.model;

import lombok.Getter;
import lombok.Setter;

@Getter
@Setter
public class User {
    private String username;
    private String password;
    private String email;
    private StatusConfirmEmail statusConfirmEmail;
}
```

AuthorizeService.java

```
package com.web.esia.service;

import com.web.esia.controller.model.LoginRequest;
import com.web.esia.controller.model.OAuthLoginResponse;
import com.web.esia.dao.ClientDao;
import com.web.esia.dao.CodeDao;
import com.web.esia.dao.UserDao;
import com.web.esia.exception.ModuleException;
import com.web.esia.model.Client;
import com.web.esia.model.Session;
import com.web.esia.model.SessionProtection;
import com.web.esia.model.User;
import lombok.RequiredArgsConstructor;
import lombok.extern.slf4j.Slf4j;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.stereotype.Service;

import javax.servlet.http.HttpServletResponse;
import java.util.Optional;
import java.util.UUID;

import static com.web.esia.controller.model.StatusResponse.BAD_REQUEST;
import static com.web.esia.controller.model.StatusResponse.GENERAL_ERROR;
import static com.web.esia.controller.model.StatusSession.INVALIDATE;
import static org.springframework.web.util.HtmlUtils.htmlEscape;

@Slf4j
@Service
@RequiredArgsConstructor(onConstructor = @__(@Autowired))
public class AuthorizeService {
    private final ClientDao clientDao;
```

```
    private final UserDao userDao;
    private final CodeDao codeDao;
    private final BCryptPasswordEncoder passwordEncoder;
    private final SessionService sessionService;

    public OAuthLoginResponse oauthLogin(LoginRequest loginRequest, String clien-
tId, HttpServletResponse response, String redirectUri) {
        LoginRequest loginRequestEscaped = new LoginRequest();
        loginRequestEscaped.setUsername(htmlEscape(loginRequest.getUsername()));
        loginRequestEscaped.setPassword(htmlEscape(loginRequest.getPassword()));
        Optional<Client> clientOpt = clientDao.fetchClient(clientId);
        if (clientOpt.isEmpty()) {
            throw new ModuleException(BAD_REQUEST);
        }
        Client client = clientOpt.get();
        if (!client.getRedirectUri().equals(redirectUri)) {
            log.error("Урл редиректа '{}' не совпадает с урлом для этого клиента
'{}'", redirectUri, client.getRedirectUri());
            throw new ModuleException(GENERAL_ERROR);
        }
        SessionProtection sessionProtection = login(loginRequestEscaped, re-
sponse);

        UUID code = UUID.randomUUID();

        if (cod-
eDao.isExistUnusedAndNotExpirationCodeFor(loginRequestEscaped.getUsername(), cli-
entId)) {
            log.error("Существует неиспользуемый и не истекший код активации для
'{}'", loginRequestEscaped.getUsername());
            throw new ModuleException(GENERAL_ERROR);
        }
        codeDao.save(code, loginRequestEscaped.getUsername(), clientId);
        return new OAuthLoginResponse(code.toString(), sessionProtec-
tion.getSessionId(), sessionProtection.getCsrfToken().toString());
    }

    public SessionProtection login(LoginRequest loginRequest, HttpServletResponse
response) {
        Optional<User> userOpt = userDao.fetchUser(loginRequest.getUsername());
        if (userOpt.isEmpty()) {
            log.error("Пользователь '{}' не найден", loginRequest.getUsername());
            throw new ModuleException(GENERAL_ERROR);
        }
        User user = userOpt.get();
        if (!passwordEncoder.matches(loginRequest.getPassword(), us-
er.getPassword())) {
            log.error("Пароли для пользователя '{}' не совпадают", us-
er.getUsername());
            throw new ModuleException(GENERAL_ERROR);
        }

        Optional<Session> session = session-
Service.fetchSessionBy(user.getUsername());
        if (session.isPresent()) {
            sessionService.invalidateSession(session.get().getSessionId());
            session.get().setStatus(INVALIDATE);
            log.info("Сессия '{} была инвалидирована для пользователя '{}'", ses-
sion.get().getSessionId(), user.getUsername());
        }

        String sessionId = sessionService.generateSessionIfNeed(null);
        UUID csrfToken = UUID.randomUUID();
        sessionService.save(user.getUsername(), sessionId, csrfToken);
```

```
        log.info("Пользователь '{}' авторизован с сессией '{}'", us-
er.getUsername(), sessionId);

        return new SessionProtection(sessionId, csrfToken);
    }

    public String oauthAuthorizeBySession(String clientId, String redirectUri,
String sessionId, String csrfToken) {
        Optional<Client> clientOpt = clientDao.fetchClient(clientId);
        if (clientOpt.isEmpty()) {
            throw new ModuleException(BAD_REQUEST);
        }
        Client client = clientOpt.get();
        if (!client.getRedirectUri().equals(redirectUri)) {
            log.error("Урл редиректа '{}' не совпадает с урлом для этого клиента
'{}'", redirectUri, client.getRedirectUri());
            throw new ModuleException(GENERAL_ERROR);
        }
        Session session = sessionService.fetchSession(sessionId);
        if (!session.getCsrfToken().equals(csrfToken)) {
            log.info("Переданный CSRF токен '{}' не соотвуетствует CSRF токену
'{}' сессии", csrfToken, session.getCsrfToken());
            throw new ModuleException(GENERAL_ERROR);
        }

        UUID code = UUID.randomUUID();

        if (codeDao.isExistUnusedAndNotExpirationCodeFor(session.getUsername(),
clientId)) {
            log.error("Существует неиспользуемый и не истекший код активации для
'{}'", session.getUsername());
            throw new ModuleException(GENERAL_ERROR);
        }
        codeDao.save(code, session.getUsername(), clientId);
        log.info("Пользователь '{}' авторизован по сессии '{}'", ses-
sion.getUsername(), session.getSessionId());
        return code.toString();
    }
}
```

ClientService.java

```
package com.web.esia.service;

import com.web.esia.controller.model.AllClientsResponse;
import com.web.esia.controller.model.ClientRequest;
import com.web.esia.controller.model.ClientResponse;
import com.web.esia.dao.ClientDao;
import lombok.RequiredArgsConstructor;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.UUID;

@Service
@RequiredArgsConstructor(onConstructor = @__(@Autowired))
public class ClientService {
    private final ClientDao clientDao;

    public ClientResponse createClient(ClientRequest clientRequest) {
```

```
        UUID clientId = UUID.randomUUID();
        UUID clientSecret = UUID.randomUUID();
        clientDao.save(clientId.toString(), clientSecret.toString(), clientRe-
quest.getName(), clientRequest.getRedirectUri());
        return new ClientResponse(clientId.toString(), clientSecret.toString());
    }

    public AllClientsResponse fetchClientsName() {
        return new AllClientsResponse(clientDao.fetchClientsName());
    }
}
```

GenerateAccessTokenService.java

```
package com.web.esia.service;

import com.auth0.jwt.JWT;
import com.auth0.jwt.algorithms.Algorithm;
import com.auth0.jwt.interfaces.DecodedJWT;
import com.web.esia.controller.model.GenerateAccessTokenRequest;
import com.web.esia.controller.model.AccessTokenResponse;
import com.web.esia.dao.GenerateAccessTokenDao;
import com.web.esia.dao.ClientDao;
import com.web.esia.dao.IdTokenDao;
import com.web.esia.exception.ModuleException;
import com.web.esia.model.Code;
import com.web.esia.dao.CodeDao;
import com.web.esia.model.StatusCode;
import lombok.RequiredArgsConstructor;
import lombok.extern.slf4j.Slf4j;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import utils.PemUtils;

import java.io.IOException;
import java.security.interfaces.RSAKey;
import java.security.interfaces.RSAPrivateKey;
import java.security.interfaces.RSAPublicKey;
import java.util.Date;
import java.util.UUID;

import static com.web.esia.controller.model.StatusResponse.BAD_REQUEST;
import static com.web.esia.controller.model.StatusResponse.GENERAL_ERROR;
import static com.web.esia.model.StatusCode.USED;

@Slf4j
@Service
@RequiredArgsConstructor(onConstructor = @__(@Autowired))
public class GenerateAccessTokenService {
    public static final long EXPIRATION_TIME = 900_000;
    private final ClientDao clientDao;
    private final CodeDao codeDao;
    private final IdTokenDao idTokenDao;
    private final GenerateAccessTokenDao generateAccessTokenDao;
    private static final String PRIVATE_KEY_FILE_RSA = "rsa-private.pem";
    private static final String PUBLIC_KEY_FILE_RSA = "rsa-public.pem";

    public AccessTokenResponse generateAccessToken(GenerateAccessTokenRequest gen-
erateAccessTokenRequest) {
        if (!clientDao.isExistClientIdWithClientSecret(
```

```
                generateAccessTokenRequest.getClientId(),
                generateAccessTokenRequest.getClientSecret())
        ) {
            log.error("Нет клиента с clientId '{}' и secretCode '{}'",
                    generateAccessTokenRequest.getClientId(),
                    generateAccessTokenRequest.getClientSecret());
            throw new ModuleException(BAD_REQUEST);
        }
        Code fetchedCode = codeDao.getCode(
                generateAccessTokenRequest.getClientId(),
                generateAccessTokenRequest.getCode()
        );
        Date now = new Date();
        boolean isExpiration = now.after(fetchedCode.getExpiration());
        if (isExpiration) {
            log.error("Код '{}' протух", generateAccessTokenRequest.getCode());
            throw new ModuleException(BAD_REQUEST);
        }
        Algorithm algorithmRS;
        try {
            RSAPrivateKey privateKey = (RSAPrivateKey) Pem-
Utils.readPrivateKeyFromFile(PRIVATE_KEY_FILE_RSA, "RSA");
            RSAPublicKey publicKey = (RSAPublicKey) Pem-
Utils.readPublicKeyFromFile(PUBLIC_KEY_FILE_RSA, "RSA");
            algorithmRS = Algorithm.RSA256(publicKey, privateKey);
        } catch (IOException e) {
            log.error("Произошла ошибка при чтении из файла приавтного ключа", e);
            throw new ModuleException(GENERAL_ERROR);
        }

        String idToken = JWT.create()
                .withSubject(fetchedCode.getUsername())
                .withExpiresAt(new Date(System.currentTimeMillis() + EXPIRA-
TION_TIME))
                .sign(algorithmRS);

        idTokenDao.save(fetchedCode.getUsername(), idToken);
        codeDao.setStatus(USED, fetchedCode.getCode().toString());
        UUID uuid = UUID.randomUUID();
        generateAccessTokenDao.save(uuid, fetchedCode.getUsername());
        return new AccessTokenResponse(uuid, idToken, "Bearer");
    }
}
```

IdTokenService.java

```
package com.web.esia.service;

import com.web.esia.controller.model.UserInfoResponse;
import com.web.esia.dao.CodeDao;
import com.web.esia.dao.GenerateAccessTokenDao;
import com.web.esia.dao.IdTokenDao;
import com.web.esia.exception.ModuleException;
import com.web.esia.model.AccessToken;
import com.web.esia.model.IdToken;
import lombok.RequiredArgsConstructor;
import lombok.extern.slf4j.Slf4j;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
```

```
import java.util.Optional;

import static com.web.esia.controller.model.StatusResponse.GENERAL_ERROR;

@Slf4j
@Service
@RequiredArgsConstructor(onConstructor = @__(@Autowired))
public class IdTokenService {
    private final GenerateAccessTokenDao generateAccessTokenDao;
    private final IdTokenDao idTokenDao;

    public UserInfoResponse fetchUserInfo(String authorization) {
        String accessTokenReq = authorization.substring(7);
        Optional<AccessToken> accessTokenOptional =
                generateAccessTokenDao.fetchAccessToken(accessTokenReq);
        if (accessTokenOptional.isEmpty()) {
            log.error("Не найдено Access Token '{}'", accessTokenReq);
            throw new ModuleException(GENERAL_ERROR);
        }
        AccessToken accessToken = accessTokenOptional.get();

        Optional<IdToken> idTokenOpt = idTo-
kenDao.fetchIdToken(accessToken.getUsername());
        if (idTokenOpt.isEmpty()) {
            log.error("ID_TOKEN для Access Code '{}' не найден", ac-
cessToken.getAccessToken());
            throw new ModuleException(GENERAL_ERROR);
        }
        IdToken idToken = idTokenOpt.get();
        log.info("Получен ID_TOKEN '{}' по ACCESS TOKEN '{}'", idToken, ac-
cessToken);
        return new UserInfoResponse(idToken.getIdToken());
    }
}
```

ResourceService.java

```
package com.web.esia.service;

import com.web.esia.controller.model.ResourceResponse;
import com.web.esia.controller.model.UserChangeRequest;
import com.web.esia.dao.GenerateAccessTokenDao;
import com.web.esia.dao.ResourceDao;
import com.web.esia.exception.ModuleException;
import com.web.esia.model.AccessToken;
import com.web.esia.model.Resource;
import com.web.esia.model.Session;
import lombok.RequiredArgsConstructor;
import lombok.extern.slf4j.Slf4j;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.Optional;

import static com.web.esia.controller.model.StatusResponse.GENERAL_ERROR;

@Slf4j
@Service
@RequiredArgsConstructor(onConstructor = @__(@Autowired))
public class ResourceService {
```

```
    private final ResourceDao resourceDao;
    private final GenerateAccessTokenDao accessTokenDao;
    private final SessionService sessionService;
    private final SessionAuthService sessionAuthService;

    public ResourceResponse getResource(String sessionId, String csrfToken) {
        sessionAuthService.sessionAuthorize(sessionId, csrfToken);
        Session session = sessionService.fetchSession(sessionId);
        Resource resource = resourceDao.fetchResource(session.getUsername());
        return new ResourceResponse(resource.getUsername(), resource.getCity(),
resource.getName(), resource.getHobby());
    }

    public ResourceResponse getResourceByAccessToken(String accessToken) {
        if (!accessToken.startsWith("Bearer ")) {
            log.error("Некорректный Access Token '{}'", accessToken);
            throw new ModuleException(GENERAL_ERROR);
        }
        String accessTokenWithoutVBearer = accessToken.substring(7);
        Optional<AccessToken> accessTokenOpt = ac-
cessTokenDao.fetchAccessToken(accessTokenWithoutVBearer);
        if (accessTokenOpt.isEmpty()) {
            log.error("Access token '{}' не найден", accessTokenWithoutVBearer);
            throw new ModuleException(GENERAL_ERROR);
        }
        AccessToken accessTokenFull = accessTokenOpt.get();
        Resource resource = resource-
Dao.fetchResource(accessTokenFull.getUsername());
        log.info("Для Access Token'а '{}' найдены ресурсы: Имя: '{}, Город: '{}',
Хобби: '{}",
                accessTokenFull.getAccessToken(), resource.getName(), re-
source.getCity(), resource.getHobby());
        return new ResourceResponse(resource.getUsername(), resource.getCity(),
resource.getName(), resource.getHobby());
    }

    public void updateUserData(UserChangeRequest userChangeRequest, String ses-
sionId) {
        String username = sessionService.fetchSession(sessionId).getUsername();
        resourceDao.update(username, userChangeRequest);
    }

    public void saveByUsername(UserChangeRequest userChangeRequest, String
username) {
        resourceDao.save(username, userChangeRequest);
    }

}
```

SessionAuthService.java

```
package com.web.esia.service;

import com.web.esia.dao.SessionDao;
import com.web.esia.exception.ModuleException;
import com.web.esia.model.Session;
import lombok.RequiredArgsConstructor;
import lombok.extern.slf4j.Slf4j;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
```

```
import static com.web.esia.controller.model.StatusResponse.GENERAL_ERROR;
import static com.web.esia.controller.model.StatusSession.INVALIDATE;

@Slf4j
@Service
@RequiredArgsConstructor(onConstructor = @__(@Autowired))
public class SessionAuthService {
    private final SessionDao sessionDao;

    public void sessionAuthorize(String sessionId, String csrfToken) {
        if (sessionId == null || !sessionDao.isSessionExist(sessionId)) {
            log.error("Сессия '{}' не найдена", sessionId);
            throw new ModuleException(GENERAL_ERROR);
        }
        Session session = sessionDao.fetchSession(sessionId).orElseThrow(
                () -> new ModuleException(GENERAL_ERROR)
        );
        if (!session.getCsrfToken().equals(csrfToken)) {
            log.info("Переданный CSRF токен '{}' не соотвуетствует CSRF токену
'{}' сессии", csrfToken, session.getCsrfToken());
            throw new ModuleException(GENERAL_ERROR);
        }

        if (session.getStatus() == INVALIDATE) {
            log.error("Сессия '{}' находится в статусе INVALIDATE", ses-
sion.getSessionId());
            throw new ModuleException(GENERAL_ERROR);
        }
    }
}
```

SessionService.java

```
package com.web.esia.service;

import com.web.esia.controller.model.StatusSession;
import com.web.esia.dao.SessionDao;
import com.web.esia.exception.ModuleException;
import com.web.esia.model.Session;
import lombok.RequiredArgsConstructor;
import lombok.extern.slf4j.Slf4j;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.Optional;
import java.util.UUID;

import static com.web.esia.controller.model.StatusResponse.GENERAL_ERROR;

@Slf4j
@Service
@RequiredArgsConstructor(onConstructor = @__(@Autowired))
public class SessionService {
    private final SessionDao sessionDao;

    public String generateSessionIfNeed(String sessionId) {
        if (sessionId == null || sessionId.isEmpty()) {
            return UUID.randomUUID().toString();
        }
```

```
        return sessionId;
    }

    public void save(String username, String sessionId, UUID csrfToken) {
        sessionDao.save(sessionId, username, csrfToken);
    }

    public Session fetchSession(String sessionId) {
        Optional<Session> sessionOpt = sessionDao.fetchSession(sessionId);
        if (sessionOpt.isEmpty()) {
            log.error("Сессия не была найдена '{}'", sessionId);
            throw new ModuleException(GENERAL_ERROR);
        }
        return sessionOpt.get();
    }

    public Optional<Session> fetchSessionBy(String username) {
        return sessionDao.fetchSessionByUsername(username);
    }

    public void invalidateSession(String sessionId) {
        sessionDao.setStatus(sessionId, StatusSession.INVALIDATE);
    }

    public void setStatus(StatusSession status, String sessionId) {
        sessionDao.setStatus(sessionId, status);

    }
}
```

SingUpService.java

```
package com.web.esia.service;

import com.web.esia.controller.model.SingUpRequest;
import com.web.esia.controller.model.UserChangeRequest;
import com.web.esia.dao.UserDao;
import com.web.esia.exception.ModuleException;
import lombok.RequiredArgsConstructor;
import lombok.extern.slf4j.Slf4j;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.stereotype.Component;

import static com.web.esia.controller.model.StatusResponse.BAD_REQUEST;
import static com.web.esia.model.StatusConfirmEmail.NEW;

@Slf4j
@Component
@RequiredArgsConstructor(onConstructor = @__(@Autowired))
public class SingUpService {
    private final UserDao userDao;
    private final BCryptPasswordEncoder encoder;
    private final ResourceService resourceService;


    public void singUp(SingUpRequest request) {
        if (userDao.isUserExist(request.getName())) {
            log.error("Пользователь с ником '{} существует", request.getName());
            throw new ModuleException(BAD_REQUEST);
```

```
        }

        if (!request.getPassword().equals(request.getPasswordRepeat())) {
            log.error("Пароли для '{}' не совпадают", request.getName());
            throw new ModuleException(BAD_REQUEST);
        }
        String passwordEncoded = encoder.encode(request.getPassword());
        userDao.save(request.getName(), passwordEncoded, NEW);

        UserChangeRequest userChangeRequest = new UserChangeRequest();
        resourceService.saveByUsername(userChangeRequest, request.getName());
        log.info("Создан новый пользователь с ником '{}'", request.getName());
    }
}
```

EsiaApplication.java

```
package com.web.esia;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import
org.springframework.session.jdbc.config.annotation.web.http.EnableJdbcHttpSession;

@SpringBootApplication
public class EsiaApplication {

    public static void main(String[] args) {
        SpringApplication.run(EsiaApplication.class, args);
    }
}
```

PemUtils.java

```
package utils;


import org.bouncycastle.util.io.pem.PemObject;
import org.bouncycastle.util.io.pem.PemReader;

import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.security.KeyFactory;
import java.security.NoSuchAlgorithmException;
import java.security.PrivateKey;
import java.security.PublicKey;
import java.security.spec.EncodedKeySpec;
import java.security.spec.InvalidKeySpecException;
import java.security.spec.PKCS8EncodedKeySpec;
import java.security.spec.X509EncodedKeySpec;

public class PemUtils {

    private static byte[] parsePEMFile(File pemFile) throws IOException {
        if (!pemFile.isFile() || !pemFile.exists()) {
```

```
            throw new FileNotFoundException(String.format("The file '%s' doesn't
exist.", pemFile.getAbsolutePath()));
        }
        PemReader reader = new PemReader(new FileReader(pemFile));
        PemObject pemObject = reader.readPemObject();
        byte[] content = pemObject.getContent();
        reader.close();
        return content;
    }

    private static PublicKey getPublicKey(byte[] keyBytes, String algorithm) {
        PublicKey publicKey = null;
        try {
            KeyFactory kf = KeyFactory.getInstance(algorithm);
            EncodedKeySpec keySpec = new X509EncodedKeySpec(keyBytes);
            publicKey = kf.generatePublic(keySpec);
        } catch (NoSuchAlgorithmException e) {
            System.out.println("Could not reconstruct the public key, the given
algorithm could not be found.");
        } catch (InvalidKeySpecException e) {
            System.out.println("Could not reconstruct the public key");
        }

        return publicKey;
    }

    private static PrivateKey getPrivateKey(byte[] keyBytes, String algorithm) {
        PrivateKey privateKey = null;
        try {
            KeyFactory kf = KeyFactory.getInstance(algorithm);
            EncodedKeySpec keySpec = new PKCS8EncodedKeySpec(keyBytes);
            privateKey = kf.generatePrivate(keySpec);
        } catch (NoSuchAlgorithmException e) {
            System.out.println("Could not reconstruct the private key, the given
algorithm could not be found.");
        } catch (InvalidKeySpecException e) {
            System.out.println("Could not reconstruct the private key");
        }

        return privateKey;
    }

    public static PublicKey readPublicKeyFromFile(String filepath, String algo-
rithm) throws IOException {
        byte[] bytes = PemUtils.parsePEMFile(new File(filepath));
        return PemUtils.getPublicKey(bytes, algorithm);
    }

    public static PrivateKey readPrivateKeyFromFile(String filepath, String algo-
rithm) throws IOException {
        byte[] bytes = PemUtils.parsePEMFile(new File(filepath));
        return PemUtils.getPrivateKey(bytes, algorithm);
    }
}
```