МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВАСТОПОЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Институт Информационных технологий и управления в технических системах

Кафедра Информационные системы

№	Дата поступления на кафедру	Подпись отв. за регистрацию	Подпись преподавателя

ОТЧЕТ

о производственной (по получению профессиональных умений и опыта профессиональной деятельности) практике

в _____
(наименование организации)

Выполнил _____Клышко Н. А._____
(Фамилия И.О. обучающегося)
_____ИС/б-17-2-о_____
(шифр группы)
Направление / специальность 09.03.02
Информационные системы и технологии
(код, наименование)

Руководитель практики от Университета
_____
(должность)
_____
(Фамилия И.О. руководителя)

Севастополь
2020 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Платформа ЭИОС СевГУ.ру была разработана для организации дистанционного обучения в Севастопольском государственном университете. Реализация проекта «СевГУ.Конференция» как части образовательной платформы была выполнена в сжатые сроки, поэтому некоторые функции не реализованы полностью.

Основными сервисами «СевГУ.Конференции» являются корпоративный мессенджер на базе Rocket.Chat и система веб-конференций BigBlueButton. Эти сервисы были интегрированы между собой, но реализуют собственную базу учётных записей пользователей. То есть платформа не интегрирована в инфраструктуру университета на уровне аккаунтов пользователей и не взаимодействует с другими внутренними сервисами, например, системой дистанционного обучения moodle. Кроме этого, проект платформы подразумевает организацию DevOps платформы GitLab для использования её студентами и преподавателями в учебном процессе.

Таким образом, задачами проекта, реализуемого в рамках производственной практики на базе кафедры «Информационные системы» Севастопольского государственного университета, являются:

1. Конфигурирование и развёртывание единой системы аутентификации, позволяющей интегрировать различные сервисы с единой базой учётных записей;

2. развёртывание и интеграция DevOps платформы GitLab.

В данном отчёте представлены обоснования выбранного для реализации проекта ПО, а также особенности его конфигурации.

1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

1.1 Протоколы технологии единого входа

Технология единого входа (англ. Single Sign-On, SSO) — технология, при использовании которой пользователь переходит из одного раздела портала в другой, либо из одной системы в другую, не связанную с первой системой, без повторной аутентификации.

Самыми распространёнными открытыми протоколами аутентификации являются SAML и OpenID Connect (OIDC). Более современным является протокол OIDC и рекомендуется для использования в качестве протокола для SSO, хотя SAML все еще часто применяется в корпоративных средах и поддерживается большим количеством ПО.

В упрощенном виде в процессе аутентификации задействованы 3 сущности: север идентификации (OpenID Connect Provider, SAML Identity Provider), клиент (Client, User Agent), сервис (Service Provider).

Процесс получения доступа к защищённому ресурсу состоит из нескольких этапов.

На первом этапе клиент запрашивает идентификацию себя у сервера (через сервис путём запроса ресурса, в случае SAML, или напрямую у сервера, в случае OIDC). Сервер идентификации может аутентифицировать пользователя различными способами, обычно, с помощью пары логин-пароль, но также и с помощью одноразового временного кода и подобных способов. В случае успешной аутентификации сервер идентификации отдаёт клиенту токен — специальную строку, которая подтверждает его право на доступ к запрошенному ресурсу.

На втором этапе клиент используя полученный токен запрашивает у сервиса необходимый ресурс. В зависимости от протокола, сервис по разно-

му проверяет токен на подлинность и в случае успешной проверки, даёт доступ к запрашиваемому ресурсу.

1.2. Сравнение альтернатив SSO Identity Provider и выбор предпочтительной для применения в рамках проекта «СевГУ.Конференция»

Одним из главных требований к ПО, которое необходимо было выбрать на данном этапе, это наличие открытых исходных кодов (Open source) и возможность бесплатного коммерческого использования.

Как уже было сказано ранее, основными протоколами, применяющимися для реализации SSO являются OIDC и SAML. Большинство программного обеспечения уже поддерживают ODIC, но при при сравнении также учитывалась поддержка протокола SAML, так как проект «СвеГУ.Конференция» должен быть интегрирован с внутренней инфраструктурой университета, в которой уже могут использоваться сервисы, не поддерживающие более современный протокол OIDC.

Формулировка задачи по выбору Identity Provider также содержала условие, по которому в качестве хранилища идентификационных данных пользователя будет использоваться служба каталогов Active Directory (AD). Поэтому, провайдер должен поддерживать интеграцию с AD или протокол LDAP.

Также, предпочтительными считались альтернативы, для которых предоставляется предварительно собранные Docker-образы для запуска сервиса в контейнере.

На основании вышеизложенных требований и сравнительной таблицы 1.1 в качестве предпочтительной реализации SSO Identity Provider был выбран проект «Keycloak». «Keycloak» удовлетворяет всем требованиям и при этом имеет наименьшие системные требования.

Таблица 1.1 – Сравнительная таблица альтернативных реализаций SSO IdP

	Keycloak	**WSO2 Identity Server**	**Gluu**	**CAS**
Поддержка OpenID Connect/ OAuth	да	да	да	да
Multi-factor authentication	да	да	да	да
Панель администратора	да	да	да	да
Поддержка протокола SAML	да	да	нет	да
Брокер учетных записей с поддержкой LDAP	да	да	-	да
Middleware	Wildfly JBOSS	WSO2 Carbon	Jetty, Apache HTTPD	Любой сервер приложений Java
Установка и настройка	простая	-	сложная	сложная
Минимальные системные требования по оперативной памяти	512 MB	2 GB	-	8 GB
Минимальные системные требования по процессору	-	2 core 1.1 Ghz	-	2 core 3 Ghz

1.3. Технология контейнеризации приложений Docker

Docker – это открытая платформа для разработки, развёртывания и запуска приложений. Docker позволяет отделить приложение от инфраструктуры, что позволяет быстро доставлять приложения. С помощью Docker, можно управлять инфраструктурой также, как происходит управление непосредственно приложениями.

Docker позволяет упаковывать и запускать приложения в слабо изолированной среде, называемой контейнером. Изоляция позволяет запускать множество контейнеров на одном сервере. Контейнеры являются легковесными, потому что им не требуются гипервизор, а они выполняются непосредственно в ядре хостовой операционной системы. Это означает, что на заданной аппаратной конфигурации можно запустить больше контейнеров, чем можно было бы запустить виртуальных машин.

Docker обеспечивает инструментарий и платформу для управления жизненным циклом контейнеров:

- Разработка приложения и его компонентов с использованием контейнера;
- Контейнер становится единицей распространения и тестирования приложения;
- Развёртывание production-приложения может производиться в локальном дата-центре, в облаке или в гибридном окружении.

1.4. Порядок конфигурации GitLab

GitLab поставляет в монолитных образах Docker, содержащих все необходимые сервисы. Для хранения постоянной информации необходимо примонтировать тома: `/var/opt/gitlab, /var/log/gitlab, /etc/gitlab.`

Конфигурация осуществляется путём редактирования файла `/etc/gitlab/gitlab.rb` или путём задания переменной среды `GITLAB_OMNIBUS_CONFIG`.

В качестве библиотеки аутентификации GitLab использует ruby-библиотеку OmniAuth. OmniAuth это стандартизированная мульти-провайдерная библиотека аутентификации для веб-приложений. Библиотека использует абстракцию, называемую «стратегия». Каждый тип провайдера аутентифика-

ции реализуется с помощью отдельного алгоритма – стратегии. Таким образом, в библиотеке достигается гибкость и простота конфигурации.

Для конфигурации стратегий OmniAuth в GitLab необходимо задать переменную `omniauth_providers`. Общая структура объекта стратегии представлена ниже:

```
{
    name: 'saml',
    args: {...},
    label: 'Company Login' # optional
}
```

Свойство `name` задаёт системное имя стратегии, которое может быть использовано в других переменных конфигурации, например, для разрешения не только аутентификации с использованием заданной стратегии но и изначальной регистрации пользователя на стороне GitLab, стратегии необходимо перечислить в переменной `omniauth_allow_single_sign_on`. Пример конфигурации регистрации с использованием 3 различных стратегий:

```
gitlab_rails['omniauth_allow_single_sign_on'] = ['saml', 'oauth2_generic', 'openid_connect']
```

Свойство `label` задаёт название стратегии, которое представлено пользователю на странице аутентификации GitLab.

В свойстве `args` в виде объекта задаются параметры стратегии, список которых зависит от класса стратегии. По-умолчанию класс определяется в зависимости от значения свойства `name`, но при необходимости можно явно задать класс стратегии с помощью свойства `strategy_class`, например:

```
strategy_class: "OmniAuth::Strategies::OAuth2Generic".
```

2 ПРАКТИЧЕСКАЯ ЧАСТЬ

2.1. Настройка LDAP-федерации в Keycloak

Для подключения Keycloak к базе учётных записей по протоколу LDAP необходимо создать новую федерацию пользователей (меню Configure > User Federation). На рисунке 2.1 изображена конфигурация федерации пользователей для работы с тестовым сервером OpenLDAP. Поля должны быть заполнены в соответствии со схемой записей в LDAP-сервисе.

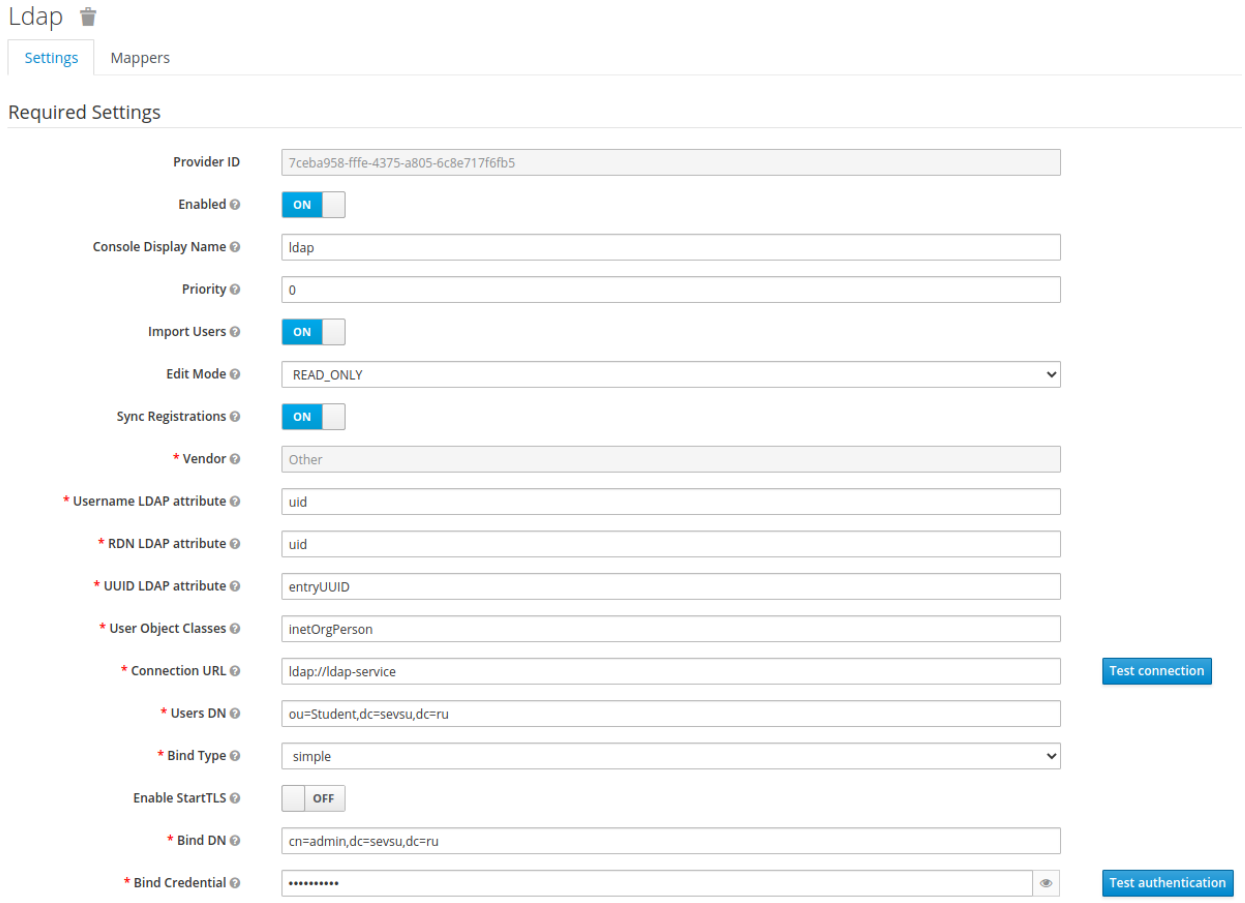

Рисунок 2.1 – Настройки федерации пользователей

Для передачи значений полей учётных записей конечным сервисам, на вкладке Mappers необходимо задать все соответствия между атрибутами

сущности LDAP и свойствами внутренней модели пользователя Keycloak. Пример конфигурации отображения изображён на рисунке 2.2.

Для дальнейшей интеграции сервиса GitLab обязательными являются поля:

- email
- username
- firstName
- lastName

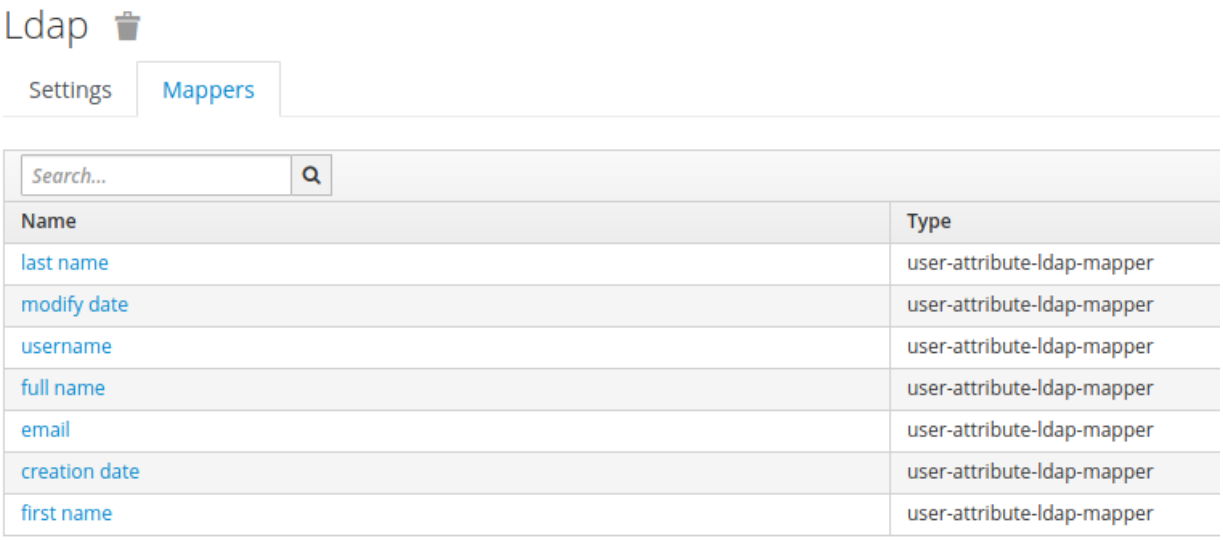

Рисунок 2.2 – Настройки отображения полей LDAP на свойства модели

2.2. Настройка OIDC-клиента в Keycloak

Для интеграции конечный сервисов с Keycloak, в панели управления необходимо создать объект клиента с соответствующим типом протокола (меню Configure > Clients).

Для ограничения доступа сторонних сервисов к аутентификации Keycloak необходимо установить свойство AccessType в значений confidential.

На рисунке 2.3 изображён пример конфигурации OIDC-клиента для интеграции с GitLab. На рисунке 2.4 изображена конфигурация отображения свойств внутренней модели на свойства пользователя GitLab.

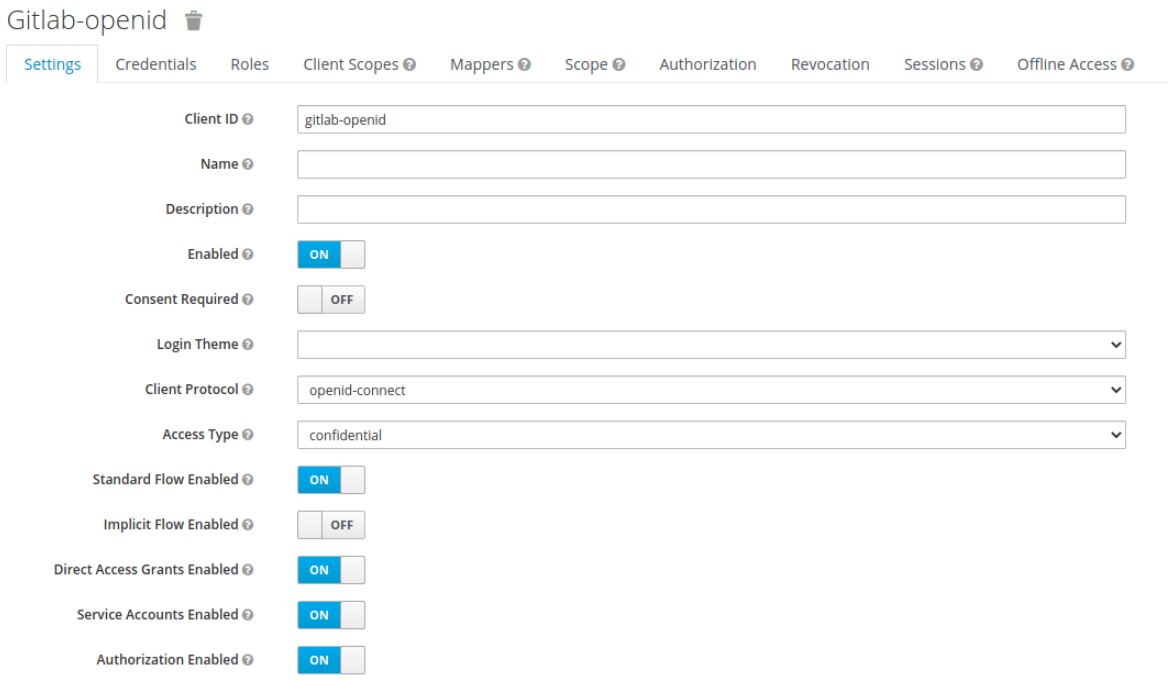

Рисунок 2.3 – Пример конфигурации OIDC-клиента для GitLab

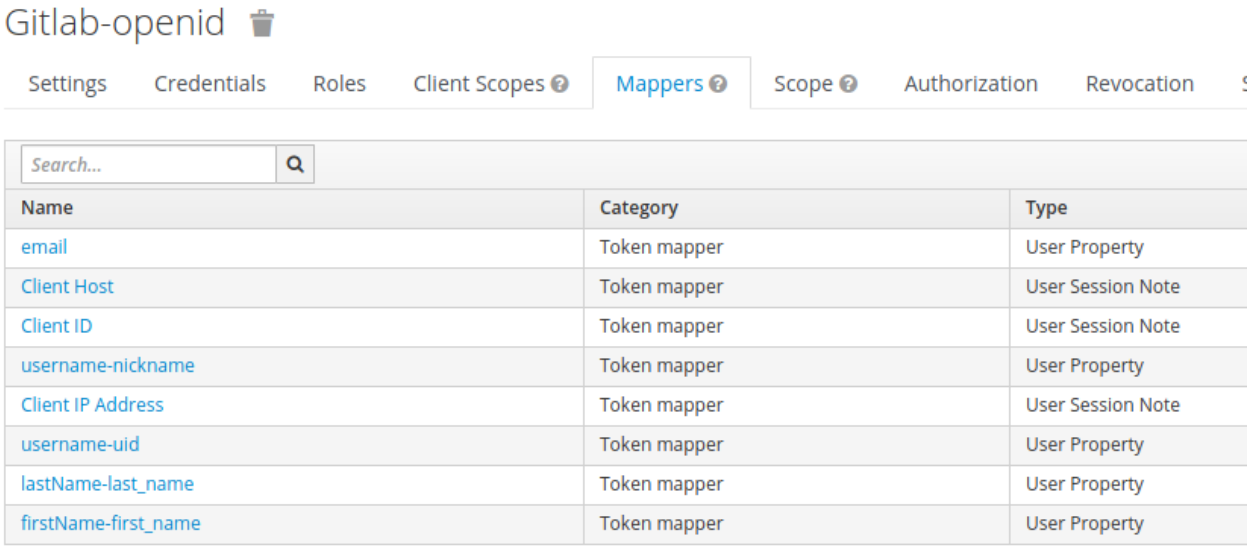

Рисунок 2.4 – Конфигурация отображения свойств внутренней модели на свойства пользователя GitLab.

2.3. Настройка GitLab для входа с помощью OIDC

Как было сказано в 1.4 — для интеграции сервиса аутентификации в GitLab необходимо сконфигурировать стратегию OmniAuth.

В случае аутентификации по протоколу OpenID Connect необходимо выполнение следующих условий:

- Свойство `issuer` задаётся как путь к корню рабочего realm'a Keycloak;
- Свойство `client_options.identifier` заполняется в соответствии с `client_id`, заданным при конфигурации Client OIDC в Keycloak;
- Свойство `client_options.secret` заполняются в соответствии с Clients > "заданный client_id" > Credentials > Secret;
- Адрес keycloak.example.com должен быть доступен из среды GitLab и иметь валидный SSL-сертификат;
- Если во время тестирования используется самоподписанный сертификат, то его необходимо поместить в `/etc/gitlab/trusted-certs` в формате pem.

Пример конфигурации OIDC-стратегии:

```
{
        name: 'openid_connect',
        label: 'OpenID',
        args: {
            name: 'openid_connect',
            scope: ['openid','profile'],
            response_type: 'code',
            issuer: 'https://keycloak.example.com:8443/auth/realms/master',
            discovery: true,
             client_auth_method: 'query',
             uid_field: 'uid',
             send_scope_to_token_endpoint: false,
             client_options: {
                 identifier: 'gitlab-openid',
                 secret: '38aab68c-0461-4a5e-a677-9c9e7952c1fe',
                 redirect_uri: 'http://gitlab.example.com/users/auth/
openid_connect/callback'
             }
         }
    }
```

Чтобы при выходе из GitLab сбрасывалась сессия пользователя Keycloak необходимо задать After sign out path в разделе Admin/Settings/ General/"Sign in restrictions" равный:

`https://keycloak.example.com:8443/auth/realms/master/protocol/openid-connect/logout?redirect_uri=https://keycloak.example.com:8443/`

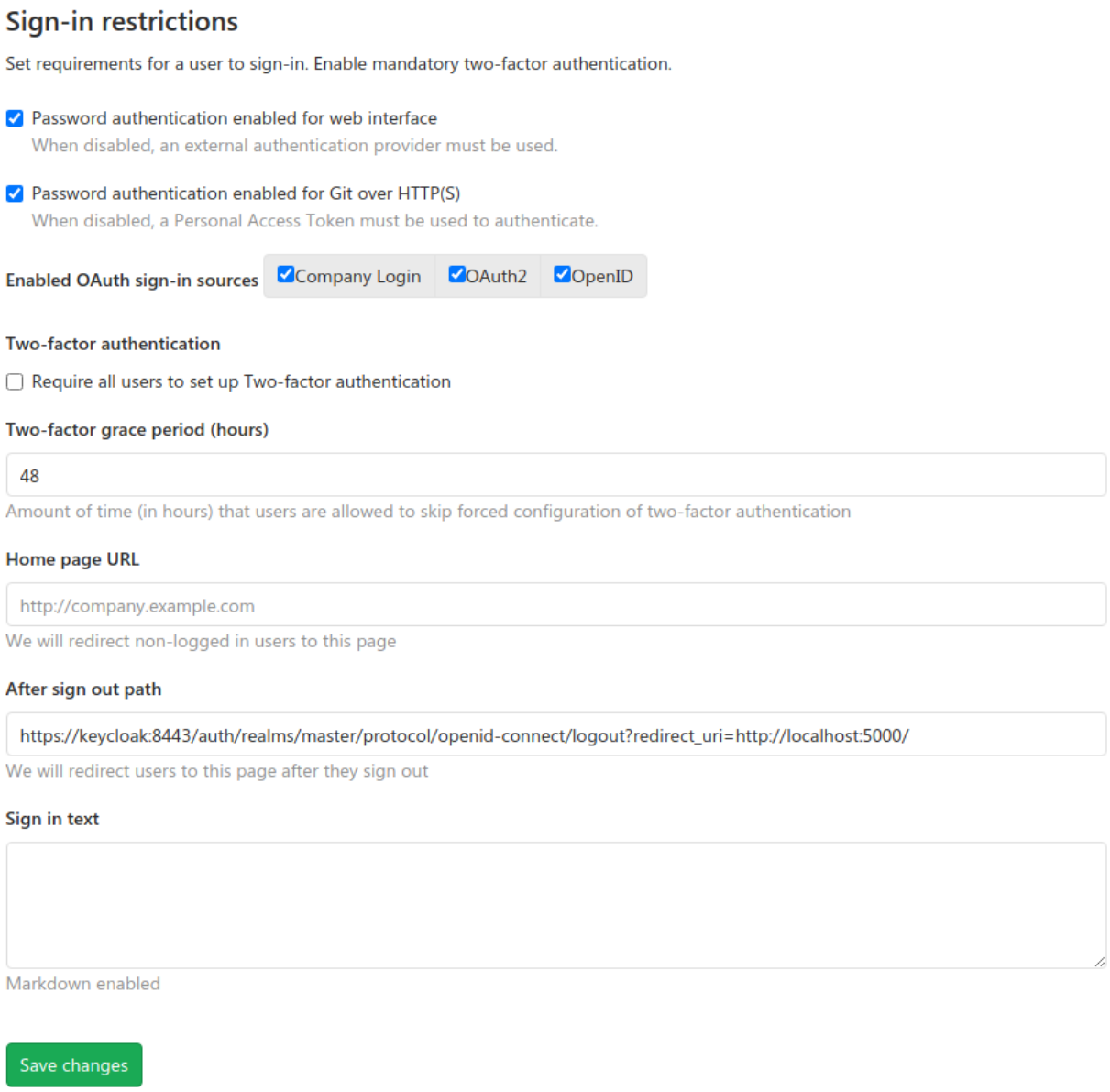

Рисунок 2.5 – Конфигурация сброса сессии при выходе пользователя из GitLab.

ЗАКЛЮЧЕНИЕ

В ходе реализации проекта в рамках прохождения производственной (по получению профессиональных умений и опыта профессиональной деятельности) практики на базе кафедры «Информационные системы» Севастопольского государственного университета были сконфигурированы сервисы Keycloak и GitLab. Для развёртывания системы была использован технология контейнеризации приложений Docker и инструмент для развертывания контейнеров DockerCompose. Для проведения локального тестирования также были сконфигурированы сервисы OpenLDAP и phpldapadmin.

Реализация данного проекта позволила приобрести навыки развёртывания, конфигурирования и администрирования приложений и сервисов промышленного уровня. Задачи в рамках проекта требовали глубокого изучения таких вопросов, как: протоколы технологии единого входа (SAML, OIDC, OAuth2), выпуск и использование SSL-сертификатов, интеграция контейнеризированных приложений.

Сконфигурированные сервисы были протестированы с использованием тестовых учётных записей. Аутентификация на сервисе GitLab через протоколы SAML, OIDC, OAuth2, предоставляемые Keycloak, работает корректно.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Administrator Docs | GitLab [Электронный ресурс]. – Электрон. текстовые дан. – 2020. - Режим доступа: https://docs.gitlab.com/ee/administration/, свободный.
2. Comparison of open-source SSO implementations [Электронный ресурс]. – Электрон. текстовые дан. – 2019. - Режим доступа: https://gist.github.com/bmaupin/6878fae9abcb63ef43f8ac9b9de8fafd, свободный.
3. Docker Documentation [Электронный ресурс]. – Электрон. текстовые дан. – 2020. - Режим доступа: https://docs.docker.com/, свободный.
4. Ldapwiki: Main [Электронный ресурс]. – Электрон. текстовые дан. – 2020. - Режим доступа: https://ldapwiki.com/wiki/Main, свободный.
5. OpenID Connect | OpenID [Электронный ресурс]. – Электрон. текстовые дан. – 2020. - Режим доступа: https://openid.net/connect/, свободный.
6. RFC 6749 - The OAuth 2.0 Authorization Framework [Электронный ресурс]. – Электрон. текстовые дан. – 2012. - Режим доступа: https://tools.ietf.org/html/rfc6749, свободный.
7. SAML Specifications | SAML XML.org [Электронный ресурс]. – Электрон. текстовые дан. – 2013. - Режим доступа: http://saml.xml.org/saml-specifications, свободный.
8. Server Administration Guide [Электронный ресурс]. – Электрон. текстовые дан. – 2020. - Режим доступа: https://www.keycloak.org/docs/latest/server_admin/index.html, свободный.

ПРИЛОЖЕНИЕ А

Исходный текст руководства по настройке readme.md

```
# Интеграция GitLab с аутентификацией Keycloak

## Настройка LDAP-федерации в Keycloak

Назначение обязательных полей вполне ясны из документации и описания в интер-
фейсе настройки

Особенности:
* Поле Connection URL должно включать также префикс протокола (например,
ldap://ldap-service.local)
* На вкладке Mappers необходимо задать все соответствия между атрибутами сущ-
ности LDAP и свойствами внутренней модели пользователя Keycloak

Для дальнейшей настройки интеграции необходимыми являются поля: email,
username, firstName, lastName

## Настройка OIDC Client в Keycloak

Настройки:
* Client Protocol: openid-connect
* Access Type: confidential # для доступа к OIDC с секретным ключом
* Valid Redirect URIs: перечислить маски для адресов, на которые будет запра-
шиваться перенаправление при аутентификации
* Admin URL: адрес, на который будут направляться backchannel logout запросы
при выходе пользователя из системы

На вкладке Mappers необходимо настроить отображение следующих свойств модели:
* email -> email # GitLab идентифицирует пользователя по email
* username -> uid # еще один уникальный идентификатор пользователя
* username -> nickname # GitLab использует это поле для генерации внутреннего
адреса пользователя при регистрации на своей стороне: nickname@gitlab.domain
* lastName -> last_name
* firstName -> first_name

Полное имя пользователя будет отображаться в GitLab в виде "first_name
last_name"

## Настройка GitLab для входа с помощью OIDC

В свойстве issuer задается путь к корню рабочего realm'а. Если задано свой-
ство discovery: true, то все необходимые endpoint'ы будут автоматически полу-
чены по адресу $issuer/.well-known/openid-configuration


Свойство client_options.identifier заполняется в соответствии с client_id,
заданным при конфигурации Client OIDC в Keycloak.
Свойство client_options.secret заполняются в соответствии с Clients > "задан-
ный client_id" > Credentials > Secret
```

Адрес keycloak.example.com должен быть доступен из среды GitLab и иметь валидный SSL-сертификат

Если во время тестирования используется самоподписанный сертификат, то его необходимо поместить в /etc/gitlab/trusted-certs в формате pem

Домен, на который выдан сертификат должен соответствовать keycloak.example.com

```
    {
        name: 'openid_connect',
        label: 'OpenID',
        args: {
            name: 'openid_connect',
            scope: ['openid','profile'],
            response_type: 'code',
            issuer: 'https://keycloak.example.com:8443/auth/realms/master',
            discovery: true,
            client_auth_method: 'query',
            uid_field: 'uid',
            send_scope_to_token_endpoint: false,
            client_options: {
                identifier: 'gitlab-openid',
                secret: '38aab68c-0461-4a5e-a677-9c9e7952c1fe',
                        redirect_uri:  'http://gitlab.example.com/users/auth/
openid_connect/callback'
            }
        }
    }
```

Чтобы при выходе из GitLab сбрасывалась сессия пользователя Keycloak необходимо задать After sign out path в разделе Admin/Settings/General/"Sign in restrictions" равный

```
        https://keycloak.example.com:8443/auth/realms/master/protocol/openid-
connect/logout?redirect_uri=https://keycloak.example.com:8443/
```

ПРИЛОЖЕНИЕ Б

Исходный текст файла docker-compose.yml

```
version: '3'
services:
  ldap-service:
    image: osixia/openldap:1.1.8
    ports:
      - '389:389'
      - '636:636'
    environment:
      LDAP_BASE_DN: 'dc=sevsu,dc=ru'
      LDAP_ORGANISATION: 'SevSU'
      LDAP_DOMAIN: 'sevsu.ru'
      LDAP_ADMIN_PASSWORD: 'admin'
    volumes:
      - '$LDAP_HOME/data:/var/lib/ldap'
      - '$LDAP_HOME/config:/etc/ldap/slapd.d'
  ldap-admin:
    image: osixia/phpldapadmin:0.9.0
    environment:
      PHPLDAPADMIN_LDAP_HOSTS: ldap-service
      PHPLDAPADMIN_HTTPS: 'false'
    ports:
      - '8082:80'
  postgres:
    image: postgres
    environment:
      POSTGRES_DB: keycloak
      POSTGRES_USER: keycloak
      POSTGRES_PASSWORD: postgres
    volumes:
      - '$PG_HOME/data:/var/lib/postgresql/data'
  keycloak:
    image: quay.io/keycloak/keycloak:10.0.2
    environment:
      KEYCLOAK_USER: admin
      KEYCLOAK_PASSWORD: admin
      DB_ADDR: postgres
      DB_VENDOR: postgres
      DB_USER: keycloak
      DB_PASSWORD: postgres
      KEYCLOAK_LOGLEVEL: TRACE
    ports:
      - '8080:8080'
      - '8443:8443'
    volumes:
      - '$KEYCLOAK_HOME/tls.crt:/etc/x509/https/tls.crt'
      - '$KEYCLOAK_HOME/tls.key:/etc/x509/https/tls.key'
  gitlab:
      image: 'gitlab/gitlab-ee:latest'
      restart: always
      ports:
          - '80:80'
          - '822:22'
      environment:
        DEBUG: 'true'
        GITLAB_OMNIBUS_CONFIG: |
            gitlab_rails['omniauth_allow_single_sign_on'] = ['saml', 'oauth2_generic',
'openid_connect']
          gitlab_rails['omniauth_block_auto_created_users'] = false
          gitlab_rails['omniauth_auto_link_ldap_user'] = true
```

```
                gitlab_rails['omniauth_providers'] = [
                  {
                    name: 'saml',
                    args: {
                           assertion_consumer_service_url: 'http://localhost/users/auth/saml/
callback',
                                                                                idp_cert_fingerprint:
'89:55:7D:32:74:30:F1:46:0C:1B:B3:13:C0:16:6A:B1:BF:40:24:A3',
                                idp_sso_target_url: 'https://localhost:8443/auth/realms/master/
protocol/saml',
                      issuer: 'gitlab.local',
                                  name_identifier_format: 'urn:oasis:names:tc:SAML:2.0:nameid-
format:persistent'
                    },
                     label: 'Company Login' # optional label for SAML login button, defaults
to "Saml"
                  },
                  {
                    name: 'oauth2_generic',
                    app_id: 'gitlab-openid',
                    app_secret: '38aab68c-0461-4a5e-a677-9c9e7952c1fe',
                    label: 'OAuth2',
                    args: {
                      client_options: {
                        site: 'https://keycloak:8443', # including port if necessary
                           user_info_url: 'https://keycloak:8443/auth/realms/master/protocol/
openid-connect/userinfo',
                           authorize_url: 'https://keycloak:8443/auth/realms/master/protocol/
openid-connect/auth',
                        token_url: 'https://keycloak:8443/auth/realms/master/protocol/openid-
connect/token'
                      },
                      user_response_structure: {
                        root_path: [],
                        attributes: {}
                      },
                      redirect_url: 'http://localhost/users/auth/oauth2_generic/callback',
                    }
                  },
                  {
                    name: 'openid_connect',
                    label: 'OpenID',
                    args: {
                      name: 'openid_connect',
                      scope: ['openid','profile'],
                      response_type: 'code',
                      issuer: 'https://keycloak:8443/auth/realms/master',
                      discovery: true,
                      client_auth_method: 'query',
                      uid_field: 'uid',
                      send_scope_to_token_endpoint: false,
                      client_options: {
                        identifier: 'gitlab-openid',
                        secret: '38aab68c-0461-4a5e-a677-9c9e7952c1fe',
                        redirect_uri: 'http://localhost/users/auth/openid_connect/callback'
                      }
                    }
                  }
                ]
            volumes:
                - '$GITLAB_HOME/config:/etc/gitlab'
                - '$GITLAB_HOME/logs:/var/log/gitlab'
                - '$GITLAB_HOME/data:/var/opt/gitlab'
```