

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
Факультет Кафедра

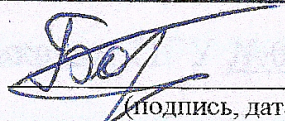
вычислительной техники

«Вычислительная техника»

Направление подготовки: 09.04.01 «Информатика и вычислительная техника»
Магистерская программа: «Вычислительные машины, комплексы, системы и сети»

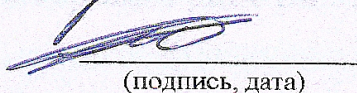
МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ
на тему
СИСТЕМА ТЕМПОРАЛЬНОГО АНАЛИЗА
МОДЕЛЕЙ ПАРАЛЛЕЛЬНЫХ
ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ

Студент


(подпись, дата)

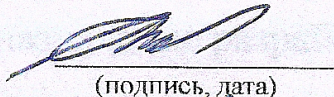
Борискин Вячеслав Владимирович
(ФИО полностью)

Руководитель


(подпись, дата)

М.П. Синева
(фамилия, инициалы)

Нормоконтролёр


(подпись, дата)

А.С. Бычков
(фамилия, инициалы)

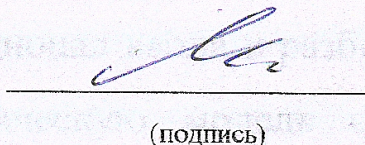
Рецензент

зав. каф. Программ. Пенз. ГУ 06.06.2022
(должность, место работы) (подпись, дата)

А.И. Мартышкин
(фамилия, инициалы)

Работа допущена к защите (протокол зас. кафедры от 02.06.2022 г. № 13)

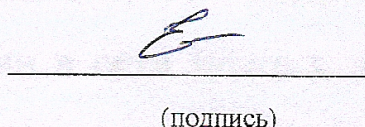
Заведующий кафедрой


(подпись)

М.А. Митрохин

Работа защищена с отметкой отлично (протокол заседания ГЭК от
14.06.2022 № 3)

Секретарь ГЭК


(подпись)

Е.И. Гурин

Пенза 2022

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет

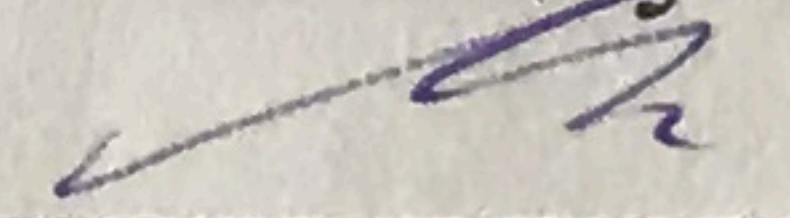
Кафедра

вычислительной техники

«Вычислительная техника»

«Утверждаю»

Заведующий кафедрой

 М.А. Митрохин

«21» сентября 2020 г.

ЗАДАНИЕ НА МАГИСТЕРСКУЮ ДИССЕРТАЦИЮ

1. Студент Борискин Вячеслав Владимирович гр.20ВВм1 факультета
вычислительной техники.

2. Тема работы: «Система темпорального анализа моделей параллельных
вычислительных систем»

Тема утверждена приказом ПГУ №656/о от "21" сентября 2020 г.

3. Руководитель магистерской работы к.т.н., доцент каф. ВТ ПГУ Синев
М.П.

4. Задание на работу (назначение разработки, исходные данные и т.п.)

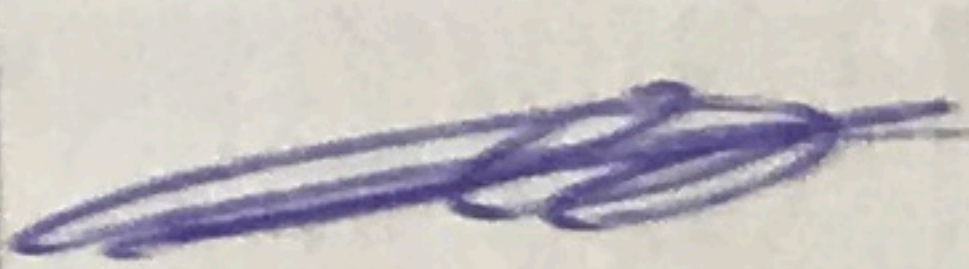
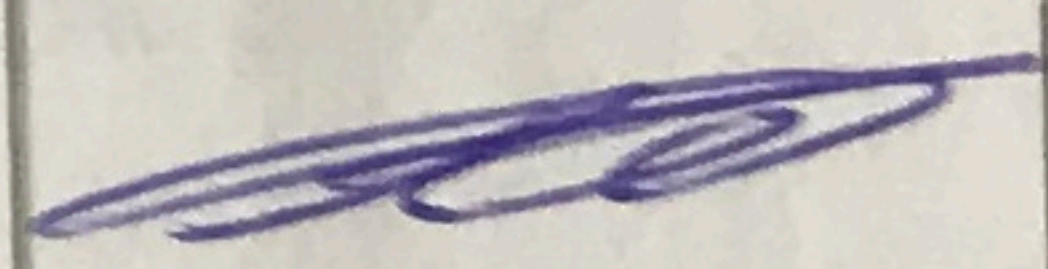
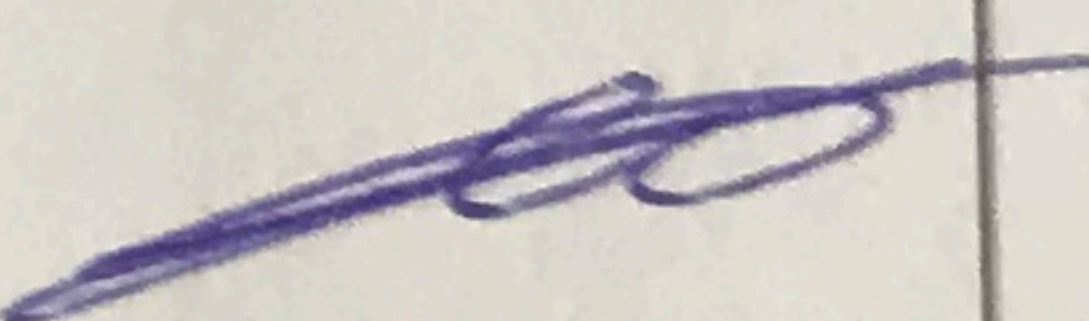
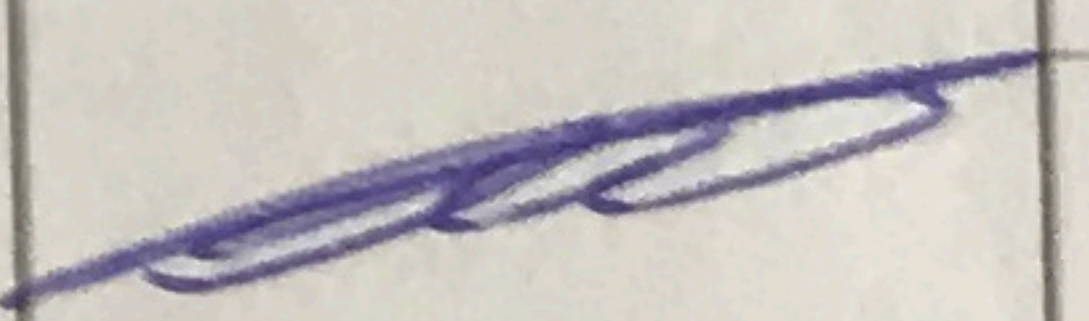
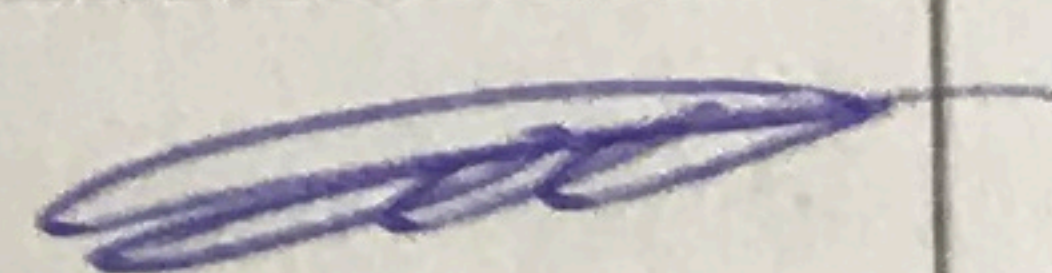
Разработать приложение «Система темпорального анализа моделей
параллельных вычислительных систем», выполняющее проектирование,
описание и моделирование поведения вычислительных систем. Приложение
должно обладать графическим пользовательским интерфейсом. Система
должна включать в себя функционал ввода и разбора системы канонических
уравнений, отображать графическую модель системы и предоставлять
пользователю возможность взаимодействовать с элементами моделируемой
системы и проводить симуляцию ее работы.

С помощью разработанной среды исследовать и промоделировать
алгоритмы защиты информации в сети Internet, такие как OAuth2, Open ID
Connect (OIDC).

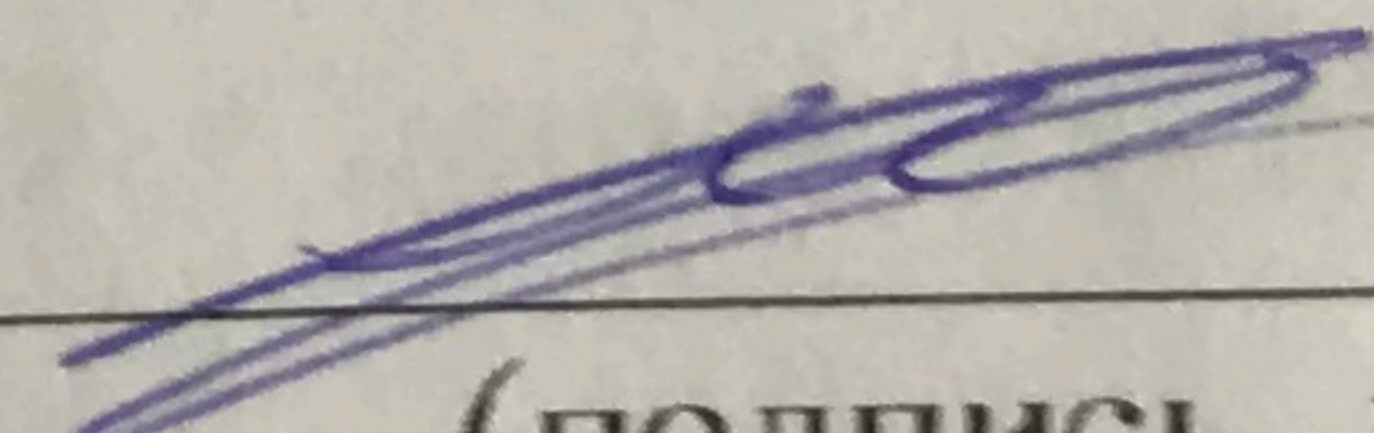
5. Перечень подлежащих разработке вопросов

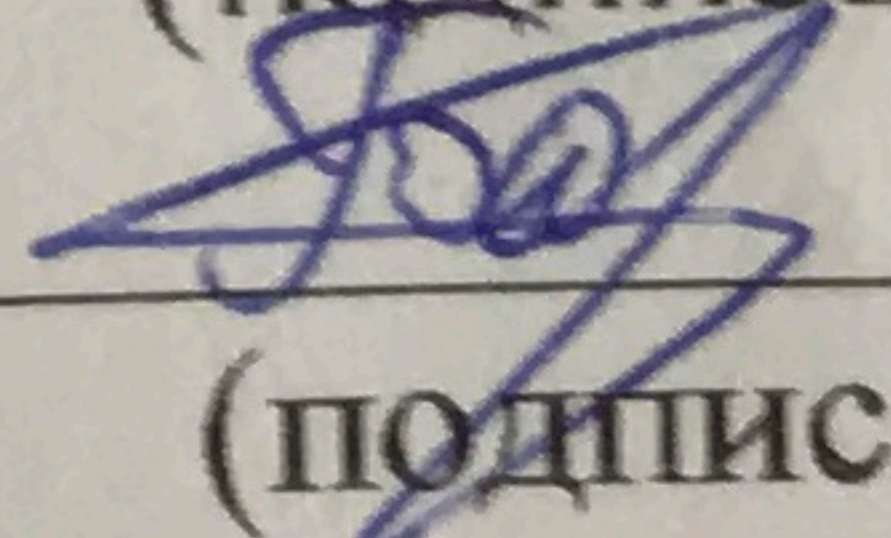
Требуется произвести анализ предметных областей: изучить актуальные на сегодняшний день методы описания информационных систем, изучить существующие методики, выполняющие описание. Также необходимо изучить современные способы и алгоритмы защиты информации в сети Internet, исследовать их принципы работы.

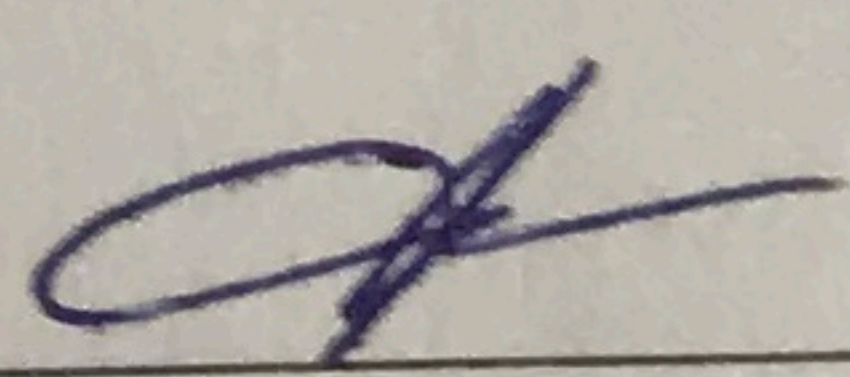
6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1.	Изучение предметной области	10%	21.09.2020 – 05.02.2021	
2.	Анализ требований	10%	06.02.2021 – 31.05.2021	
3.	Выбор технологий, применяемых при проектировании и разработке	15%	01.06.2021 – 01.08.2021	
4.	Описание алгоритмов	15%	01.08.2021 – 31.08.2021	
5.	Разработка приложения «Система темпорального анализа моделей параллельных вычислительных систем»	15%	01.09.2021 – 31.01.2022	
6.	Разработка приложения, реализующего протоколы защиты данных пользователя	25%	01.02.2022 – 20.05.2022	
7.	Оформление пояснительной записки	10%	21.05.2022 - 31.05.2022	

Дата выдачи задания "21" сентября 2020 г.

Руководитель магистерской работы  21.09.20 Синева М.П.
(подпись, дата)

Задание к исполнению принял студент  21.09.20 Борискин В.В.
(подпись, дата)

Работу к защите допустить
Декан факультета  Фионова Л.Р.
09.06.2022 (подпись, дата)

РЕФЕРАТ

Магистерская диссертация содержит 151 лист, 57 рисунков, 3 таблицы, 10 формул, 20 источников, 3 приложения.

АВТОМАТЫ, МОДЕЛИ, МОДЕЛИРОВАНИЕ, СИМУЛЯЦИЯ, ЗАЩИТА ИНФОРМАЦИИ, ПРОТОКОЛЫ ЗАЩИТЫ, OAuth2, OIDC

Цель работы – разработать систему, позволяющую проектировать, анализировать и моделировать сложные распределённые информационные системы, подразумевающие в своей работе защиту данных пользователя.

Объектом разработки является приложение, обеспечивающее выполнение следующих функций: ввод системы канонических уравнений, описывающих моделируемую систему, анализ и разбор введенной системы канонических уравнений, создание визуальной модели вычислительной системы, описанной с помощью конечного автомата, моделирование поведения вычислительной системы. Приложение должно обладать графическим пользовательским интерфейсом.

Также объектом разработки является приложение, реализующее исследуемые протоколы защиты информации OAuth2, Open ID Connect (OIDC).

В результате проведенной работы разработано приложение, удовлетворяющее поставленной задаче, а именно: разработаны алгоритмы парсинга системы канонических уравнений, отображения графических объектов временного автомата, описывающего вычислительную систему, моделирования и симуляции работы вычислительной системы, также приложение, реализующее исследуемые протоколы защиты информации OAuth2, Open ID Connect (OIDC). Приложения успешно протестированы и готовы к использованию.

ПГУ 1.090401.09.001 ПЗ

					ПГУ 1.090401.09.001 ПЗ				
Изм.	Лист	№ докум.	Подпись	Дата	Система темпорального анализа моделей параллельных вычислительных систем Пояснительная записка	Лит.	Лист	Листов	
Разраб.		Борискин В.В.							
Провер.		Синев М.А.					2	149	
Н. Контр.		Бычков А.С.				ФВТ, каф. ВТ гр. 20ВВм1			
Утверд.		Митрохин М.А.							

СОДЕРЖАНИЕ

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ	5
ВВЕДЕНИЕ.....	6
1 Анализ предметной области	7
1.1 Защита информации при передаче по сети.	7
1.2 Защита личных данных, хранящихся на сервере.....	9
1.3 Идентификационные брокеры	10
1.3.1 SSO	11
1.3.2 OAuth2	12
1.3.3 OpenId Connect (OIDC).....	17
1.4 Теория автоматов	20
1.4.1 Задачи теории автоматов.....	20
1.4.2 Классификация автоматов.....	21
1.4.3 Способы описания вычислительных систем.....	22
1.5 Цели и задачи.....	25
1.6 Анализ существующих методов моделирования	25
1.6.1 Цепи Маркова.....	25
1.6.2 Сети Петри	26
1.6.3 Конечные автоматы	27
1.7 Выбор языка программирования.....	28
1.7.1 Язык программирования C++	28
1.7.2 Язык программирования JavaScript	29
1.7.3 Язык программирования C#	29
1.7.4 Язык программирования Java	29
1.8 Выводы.....	30
2 Моделирование.....	31
2.1 Описание работы разработанной среды	31
2.2 Описание алгоритмов	33
2.2.1 Алгоритм разбора и анализа SKU	33
2.2.2 Алгоритм отрисовки компонентов.....	35

2.2.3 Алгоритм моделирования поведения системы	37
2.3 Моделирование алгоритма OAuth2	38
2.3.1 Задание КА методом SKU	39
2.3.2 Задание КА в виде графа	40
2.3.3 Задание КА с методом отмеченных таблиц	42
2.3.4 Моделирование алгоритма в разработанной среде	42
2.4 Моделирование OIDC (Open ID Connect).....	50
2.5 Выводы.....	51
3 Реализация.....	52
3.1 Keycloak.....	52
3.2 Разработка приложения	54
3.2.1 Развертывание и настройка Keycloak	54
3.2.2 Разработка клиентского приложения.....	59
3.3 Выводы.....	65
ЗАКЛЮЧЕНИЕ	66
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	67
ПРИЛОЖЕНИЕ А Лстинги среды моделирования.....	69
ПРИЛОЖЕНИЕ Б Лстинги приложения авторизации.....	126
ПРИЛОЖЕНИЕ В Презентация	132

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ

ВС – вычислительная система

ГОСТ – региональный стандарт, принятый Межгосударственным советом по стандартизации, метрологии и сертификации Содружества Независимых Государств

ЕС – Едина система

КА – конечный автомат

ПЭВМ – персональная электронно-вычислительная машина

СКУ – система канонических уравнений

ТПВ – таблица переходов и выходов

API (Application Programming Interface) – программный интерфейс приложения

OIDC (Open Id Connect) – протокол авторизации

SSO (Single Sign On / Single Sign Out) – протокол авторизации

SAML (Security Assertion Markup Language) – протокол авторизации

UML (Unified Modeling Language) – унифицированный язык моделирования

ВВЕДЕНИЕ

В современном мире процесс цифровизации все более проникает в различные сферы и аспекты жизни общества и человека, в частности. В свою очередь, как следствие подобного явления, наблюдается увеличение объемов информации используемой и передаваемой в сети Internet. Вследствие чего возникает проблема защиты информации при передаче по различным каналам сети Internet, как между пользователями напрямую (архитектура клиент-клиент), так и между пользователем и удаленным сервером (архитектура клиент-сервер).

В частности, требуют защиты информации распределенные системы – информационные системы, состоящие из некоторого количество самостоятельных юнитов, каждый из которых отвечает за ограниченный функционал в рамках целой системы. Такие части называют микросервисами. Подобный подход к архитектуре вычислительной системы позволяет сделать ее более отказоустойчивой и масштабируемой т.к. в случае отказа одного микросервиса остальные продолжают работать.

Защита нужна для того, чтобы организовать взаимодействие между микросервисами и четко различать какой пользователь является инициатором запроса к тому или иному микросервису в системе, и не отправить пользователю данные, для него не предназначенные, например личные данные другого пользователя (номер телефона, данные документов, номера банковских карт и т.д.) [1].

1 Анализ предметной области

Отправной точкой в проектировании и разработке любого приложения является анализ предметной области. Этот этап необходим для четкой постановки целей, формировании требований к конечному продукту и определении ожидаемых результатов всей работы. В этом разделе проводится обзор основных аспектов и методик описания информационных систем.

1.1 Защита информации при передаче по сети

Базовым протоколом при передаче данных между отправителем и получателем по сети является защищённый протокол передачи данных https (Hyper Text Transport Protocol Secure) [2]. Данный протокол основывается на шифровании передаваемых данных с помощью SSL или TLS сертификатов.

Принцип данного протокола заключается в использовании публичного, приватного и общего сессионного ключа шифрования данных, передаваемых между клиентом и сервером. В протоколе используется два алгоритма [3]:

1) алгоритм асимметричного шифрования RSA (Rivest, Shamir, Adleman). Суть алгоритма заключается в следующем. Имеются публичный и приватный ключи. Информация зашифровывается только публичным ключом и передается в зашифрованном виде, а расшифровывается только приватным. Ключи не взаимозаменяемые. Такой подход довольно безопасный, но медленный,

2) алгоритм симметричного шифрования Диффи-Хеллмана. Основное свойство алгоритма, в том, что информация зашифровывается и расшифровывается одним и тем ключом. Данный алгоритм работает намного быстрее асимметричного RSA, но менее безопасный.

Алгоритм обмена данными в таком случае состоит из 6 шагов.

Шаг 1. Клиент (браузер или клиентский сервер) отправляет запрос на сервер с целью выяснить, установлен ли для данного сервера сертификат.

Шаг 2. В ответ, сервер сообщает клиенту свой сертификат вместе с публичным ключом.

Шаг 3. Клиент удостоверяется, что сертификат подлинный, т.е. подписан официальным сертификационным центром.

Шаг 4. Клиент генерирует сессионный ключ, подписывает его публичным ключом, полученным от сервера, и отправляет серверу в зашифрованном виде.

Шаг 5. Сервер расшифровывает полученный сессионный ключ своим приватным ключом, который хранится в тайне и сохраняет ключ в памяти.

Шаг 6. В результате клиент и сервер имеют симметричный сессионный ключ, устанавливаемый в границах сессии, с помощью которого зашифровывают и расшифровывают передаваемую по сети информацию.

UML-диаграмма последовательности приведена на рисунке 1.

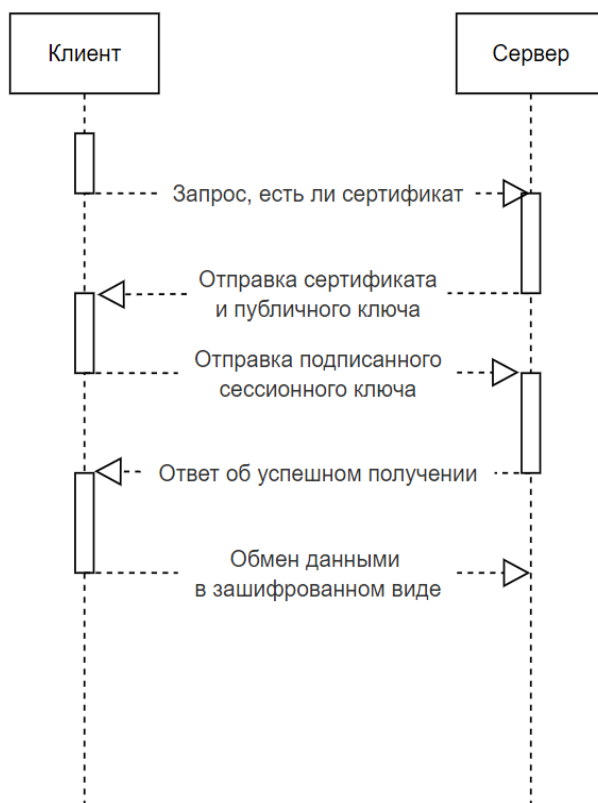


Рисунок 1 - UML последовательности https

С помощью сгенерированного сессионного ключа происходит зашифровка данных перед отправкой одной стороной и расшифровка данных после получения их другой стороной. Сессионный ключ также держится в тайне и устанавливается только в рамках одного соединения, т.е. не хранится

ни на одной стороне постоянно. В противном случае, если злоумышленник завладеет таким ключом, он может перехватить данные в процессе обмена и расшифровать их, что неприемлемо.

1.2 Защита личных данных, хранящихся на сервере

Помимо защиты во время передачи данных между двумя сторонами, необходимо также защитить данные, хранящиеся на сервере и не предназначенные не авторизованному лицу ни в каком виде, даже в зашифрованном. Это касается таких личных данных, как ФИО, телефонный номер, серии и номера документов, финансовые реквизиты и т.д. Для получения подобных данных необходимо провести идентификацию клиента, следом аутентификацию клиента, а затем его авторизацию, непосредственно при запросе защищенных данных.

При установлении доступа описанным выше методом выделяют три фазы:

1) идентификация. Выявление и установка пользователя по его идентификатору (имя, логин, электронная почта, номер телефона т. д.). Цель данной фазы определить, с каким именно пользователем будут происходить дальнейшие проверки,

2) аутентификация. Цель фазы - удостоверение, что идентифицируемый пользователь является именно тем, за кого себя выдает. Например, проверка пользователя по паролю, электронной подписи, сертификату и т.д,

3) авторизация. Установка доступа аутентифицированного пользователя к защищенному ресурсу. Цель фазы - выдача пользователю, прошедшему аутентификацию прав на действия с запрашиваемым ресурсом. UML-диаграмма последовательности приведена на рисунке 2.

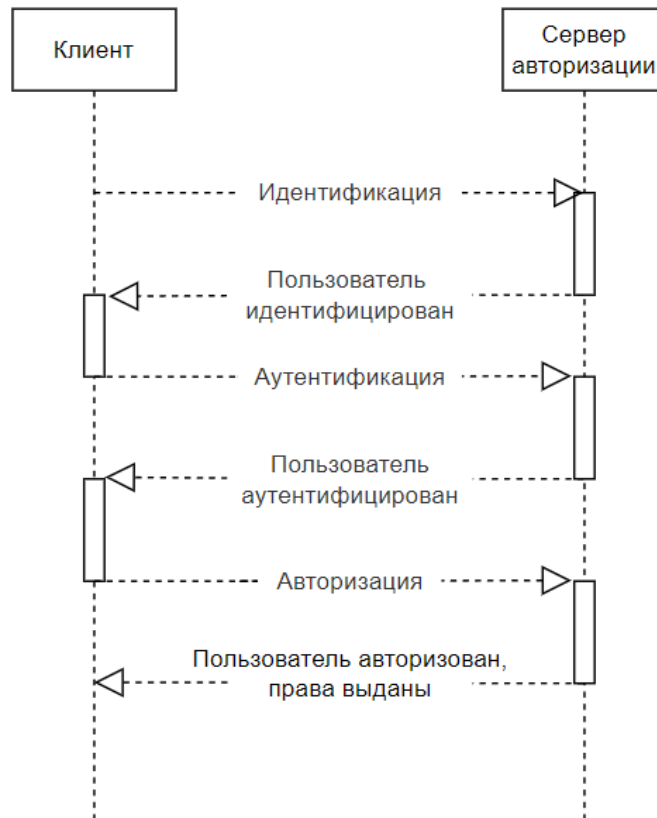


Рисунок 2 - UML-диаграмма получение доступа

Инициировать подобные проверки можно разными способами: с помощью учетных данных пользователя (логин и пароль), с помощью электронной подписи (ЭП), с помощью сертификатов, его ролей в системе и т.д. По подобному алгоритму работают большинство современных информационных систем, требующих защиту данных их пользователей.

1.3 Идентификационные брокеры

При реализации вышеописанных фаз применяют так называемые идентификационные брокеры – подключаемые библиотеки или отдельно работающие приложения, выполняющие идентификацию, аутентификацию и авторизацию пользователя. Взаимодействие приложений и идентификационных брокеров может происходить по нескольким протоколам. Самые популярные из них представлены ниже:

- SSO,
- OAuth,

- OAuth2,
- OIDC,
- SAML.

1.3.1 Протокол SSO

Большинство идентификационных брокеров поддерживают протокол единого входа SSO. Суть данной технологии заключается в следующем. Предположим, имеется система, состоящая из многих приложений и серверов, активно обменивающихся между собой информацией - микросервисов. Каждый сервер необходимо защитить от атак злоумышленников и запросов третьих лиц, не имеющих право на получения хранящейся на сервере информации.

В таком случае, всякий раз, когда внутри системы будет происходить обмен информацией между сервисами, пользователю необходимо снова подтверждать свою личность вводом пароля или другим способом. Это может быть утомительным и трудоемким процессом, который значительно упрощает спецификация SSO. Суть ее такова, что при вводе пользователем своих идентификационных данных однажды, ему больше не придется повторять это снова при внутреннем взаимодействии сервисов.

Реализуется это путем выдачи токена - строки, хранящей информацию о пользователе и подписанной секретом по выбранному алгоритму для удостоверения, что этот токен действительно выдан SSO-сервером, а не поддельный. Чаще всего токен передается в заголовках запроса к каждому микросервису.

Разумеется, вполне уместны случаи, когда пользователя необходимо обязать снова подтвердить свою личность, например при длительном бездействии. Данное поведение идентификационного брокера легко сконфигурировать, т.к. SSO предоставляет гибкий подход к настройке процессов идентификации, аутентификации и авторизации пользователей. UML-диаграмма последовательности приведена на рисунке 3.

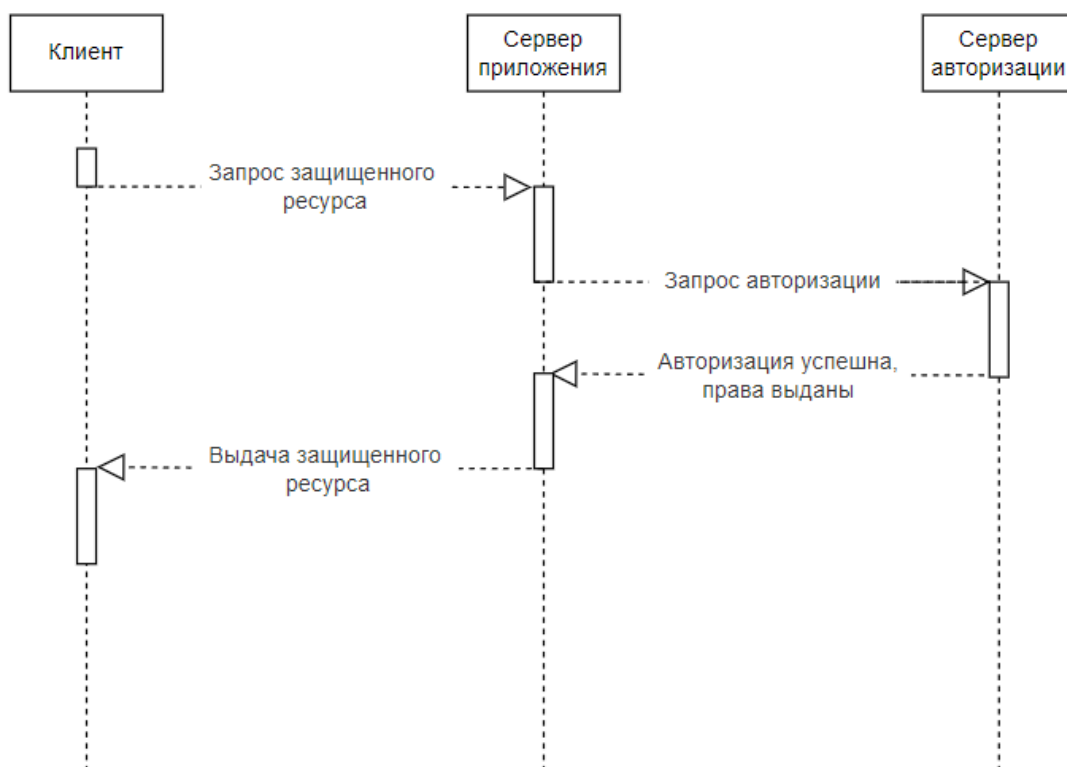


Рисунок 3 - UML-диаграмма авторизации через SSO

1.3.2 Протокол OAuth2

Также, одним из популярных и часто используемых протоколов при взаимодействии приложений с защищенными ресурсами является протокол OAuth2 [4], являющийся логическим продолжением OAuth 1.0. Основное отличие этого протокола от его более ранней версии заключается в том, что его поведение проще конфигурировать во время прикладного использования. Задача протокола - позволить выдать одному сервису (приложению) права на доступ к ресурсам пользователя на другом сервисе. Общая схема работы протокола состоит из следующих шагов:

- получение авторизации,
- обращение к защищенным ресурсам.

Результатом авторизации является access token, предъявление которого необходимо при доступе к защищенным ресурсам. Обращение к ним в самом простом случае происходит по HTTPS с указанием access token'a в заголовках или в качестве одного из параметров запроса.

В протоколе описано несколько вариантов авторизации, подходящих для различных ситуаций:

- авторизация для приложений, имеющих серверную часть (чаще всего, это сайты и веб-приложения),
- авторизация для полностью клиентских приложений (мобильные и desktop-приложения),
- авторизация по логину и паролю.

Общая схема работы протокола приведена на рисунке 4.

Абстрактное описание протокола



Рисунок 4 - OAuth2-авторизация

Авторизация приложений с серверной частью. При таком варианте использования, приложения запрашивает доступ к ресурсам пользователя другого приложения по следующему алгоритму, состоящему из 5 шагов.

Шаг 1. разрабатываемое приложение должно зарегистрироваться в стороннем приложении для получения своих уникальных `clientId`, и `clientSecret`. Эти данные необходимы при авторизации клиентского приложения в стороннем.

Шаг 2. клиентское приложение запрашивает разрешение на авторизацию, сообщая при этом свой `clientId`, `redirect_uri` – адрес куда следует перенаправить пользователя после авторизации, `scope` – область распространения прав управления данными пользователя в клиентском приложении, например только на чтение данных. Пример запроса: `POST https://<host>:<port>/oauth/authorize?response_type=code&client_id=CLIENT_ID&redirect_uri=CALLBACK_URL&scope=read`.

Шаг 3. если пользователь не авторизован, стороннее приложение должно запросить доступ пользователя к своей учетной записи. После успешной авторизации пользовательский агент (браузер) выполняет перенаправление (`redirect`) на адрес, указанный в `redirect_uri`, добавляя к запросу дополнительный параметр `code`, случайное число, необходимое для последующего запрашивания токена доступа (`access_token`).

Шаг 4. после перенаправления, описанного в предыдущем шаге, клиентское приложение запрашивает токен доступа у стороннего, предоставляя полученный код и свои данные `clientId` и `clientSecret`, а также `redirect_uri` и тип авторизации. Пример запроса: `https://<host>:<port>/oauth/token?client_id=CLIENT_ID&client_secret=CLIENT_SECRET&grant_type=authorization_code&code=AUTHORIZATION_CODE&redirect_uri=CALLBACK_URL`. Если данные корректны, то стороннее приложение возвращает в ответе токен доступа, а также выполняет перенаправление на указанный адрес. Пример ответа представлен на рисунке 5.

Шаг 5. с этого момента приложение авторизовано и может выполнять авторизованные запросы на сторонний сервис для взаимодействия с пользовательскими данными в этом сервисе.

Схема работы авторизации приложений с серверной частью представлена на рисунке 6.

```

{
  "access_token": "ACCESS_TOKEN",
  "token_type": "bearer",
  "expires_in": 2592000,
  "refresh_token": "REFRESH_TOKEN",
  "scope": "read",
  "uid": 100101,
  "info": {
    "name": "Mark E. Mark",
    "email": "mark@thefunkybunch.com"
  }
}

```

Рисунок 5 - Получение токена доступа

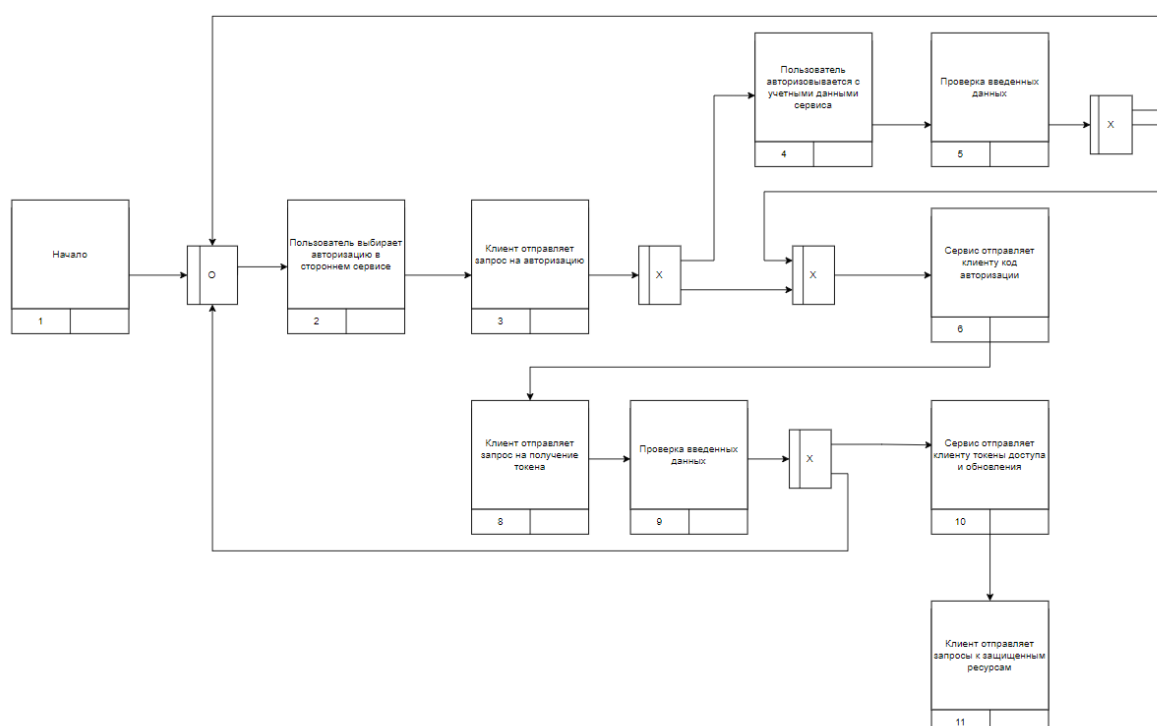


Рисунок 6 – IDEF3 модель авторизации приложений с серверной частью

Авторизация клиентских приложений. Неявный тип авторизации, при котором стороннее приложение возвращает сразу токен доступа, без кода авторизации. Такой подход менее безопасен, чем приведенный выше т.к. где конфиденциальность секрета клиента (clientSecret) не может быть гарантирована. Данный тип авторизации подходит для приложения с полностью клиентской частью, работающих в браузере, либо десктопных приложений. Алгоритм состоит из 5 шагов.

Шаг 1. клиентское приложение запрашивает доступ у стороннего, по аналогичному с предыдущим типом запросом, за исключением того, что параметр запроса `response_type=token`. Пример запроса: `POST https://<host>:<port>/oauth/authorize?response_type=token&client_id=CLIENT_ID&redirect_uri=CALLBACK_URL&scope=read`.

Шаг 2. Далее, пользователь авторизуется в стороннем сервисе, предоставляя свои учетные данные, если он этого уже не сделал ранее.

Шаг 3. После авторизации на стороне клиента происходит перенаправление (`redirect`) по адресу, указанному в параметре запроса из первого пункта `redirect_uri`, при этом сторонний сервис добавляет к `uri` дополнительный параметр `token`.

Шаг 4. По адресу `redirection_uri` должен располагаться скрипт, извлекающий токен доступа из запроса, и сохраняющий его в свое адресное пространство для дальнейшего использования.

Шаг 5. С этого момента приложение авторизовано и может выполнять авторизованные запросы на сторонний сервис для взаимодействия с пользовательскими данными в этом сервисе. Схема работы авторизации клиентских приложений представлена на рисунке 7.

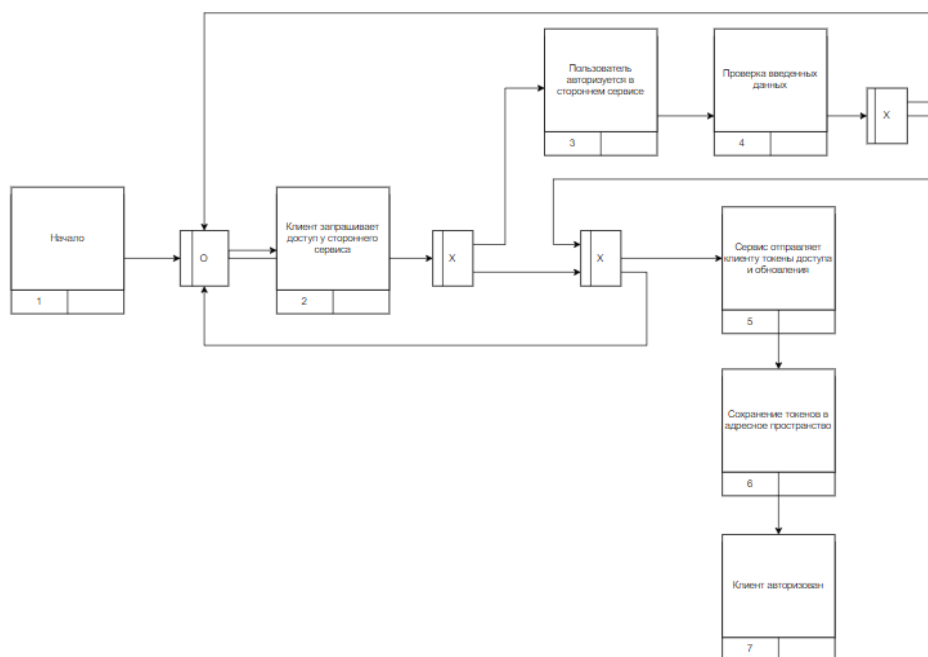


Рисунок 7 – IDEF3 модель авторизации клиентских приложений

Авторизация с предоставлением учетных данных клиента. Данный тип авторизации уступает по безопасности предыдущим двум типам, ввиду того что клиентское приложение предоставляет свои, сервисные, учетные данные в стороннем сервисе. Такие же логин и пароль, только для аутентификации учетной записи сервиса, а не пользователя. В таком случае на стороне стороннего приложения требуется создать учетную запись для клиентского приложения и определить и настроить доступ для данных, разрешенных для использования клиентским сервисом. Схема работы авторизации с предоставлением учетных данных клиента представлена на рисунке 8.

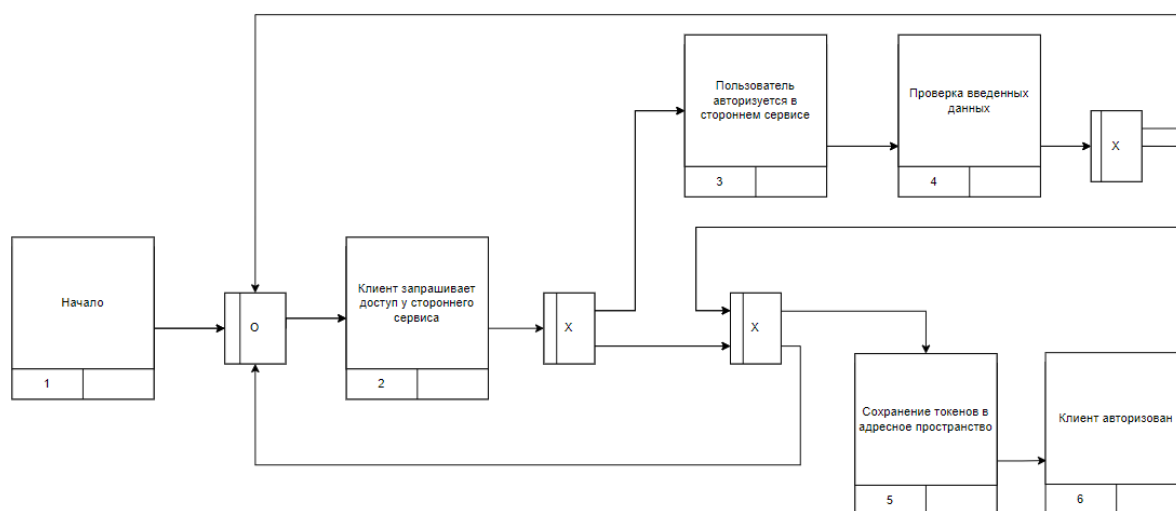


Рисунок 8 - IDEF3 модель авторизации с предоставлением учетных данных клиента

1.3.3 Протокол OIDC

Еще один популярный протокол, выполняющий авторизацию пользователя. Описанный выше протокол OAuth2.0 отвечает непосредственно за аутентификацию пользователя [5], т.е. за удостоверение его по паролю, электронной подписи, USB-ключа, сертификата и т.д. OpenId Connect служит для целей распознавания пользователя. Иными словами, можно сказать, протокол OAuth2 применяется для того, чтобы определить, имеет ли конкретный пользователь доступ к защищенным ресурсам сервера. Open Id Connect же служит для того, чтобы понять, какой именно конкретный пользователь имеет доступ к защищенным ресурсам.

Технически, OIDC работает как надстройка над протоколом OAuth2, т.е. основан поверх него. Работа происходит по алгоритму из 4 шагов [6].

Шаг 1. Также, как и в случае с OAuth2, необходимо отправить POST-запрос на сервер аутентификации (Authentication Provider), при этом в параметр запроса scope необходимо добавить тип области видимости openid. Пример запроса: POST `https:<host>:<port>/oauth/token?response_type=token%20id_token&scope=openid&redirect_uri=<redirect_uri>`.

Шаг 2. Также, как и в OAuth2, запрашивается авторизация пользователя. Если она прошла успешно, пользователю возвращается код авторизации `authorization_code`.

Шаг 3. Аналогично происходит POST запрос на сервер авторизации, для получения токенов.

Шаг 4. Вместе с токенами доступа и обновления (`access_token`, `refresh_token`), возвращается еще один токен, называемый `id_token`.

`Id_token` содержит информацию о пользователе в формате json, но в закодированном виде. Алгоритм кодировки обычно Base64Url. Т.е. данные о пользователе сохраняются в json объект, который следом кодируется в строку, которая возвращается клиенту в виде `id_token`. Данный метод передачи информации называется JWT (Json Web Token). Фактически, jwt-токен это строка, состоящая из трех частей, разделенных точками.

Первая часть описывает алгоритм кодирования токена и его тип. Пример представлен на рисунке 9.



```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

Рисунок 9 - Описание алгоритма кодирования токена и его тип

Вторая часть содержит непосредственно полезную нагрузку (`payload`), представляющую собой информацию о пользователе. Пример на рисунке 10.

```
{  
  "sub": "1234567890",  
  "name": "John Doe",  
  "admin": true  
}
```

Рисунок 10 - Payload

Эта спецификация определяет набор зарезервированных полей, которые можно расширять собственными для предоставления большей информации. С точки зрения спецификации JWT, каждое поле в json называется заявкой (claim). Список обязательных полей приведен ниже:

- iss (issuer),
- exp (expiration time),
- sub (subject),
- aud (audience).

Третья часть является зашифрованной подписью токена, необходимая для проверки подлинности токена на стороне сервера авторизации. Алгоритм шифрования указан в первой части токена и работает в соответствии с формулой, представленной на рисунке 11:

```
 HMACSHA256(  
   base64UrlEncode(header) + "." +  
   base64UrlEncode(payload),  
   secret)
```

Рисунок 11 - Шифрованная подпись токена

где, 1 часть (header) и 2 часть (payload) кодируются алгоритмом Base64Url, конкатенируются вместе и зашифровываются выбранным алгоритмом шифрования с использованием секрета, строки с набором символов, являющейся ключом к проверке подписи. При получении jwt токена, сервер авторизации проверяет его подпись помощью указанного секрета, и если фактический результат не совпадает с ожидаемым, сервер отклоняет запрос на доступ к защищенным ресурсам.

Также OIDC предусматривает наличие дополнительного эндпоинта (обычно /userinfo) для получения информации о пользователе при предъявлении access_token. Также эту информацию можно получить из второй части jwt токена.

Важно понимать, что информация в jwt-токене (payload) никаким образом не зашифровывается, за исключением подписи токена, а просто кодируются. Следовательно лицам, несогласованно получившим доступ к jwt-токену не составит труда прочесть из него информацию. Поэтому категорически не рекомендуется кодировать в токен секретные данные, такие как логины, пароли, номера банковских карт, данные документов и прочее.

1.4 Теория автоматов

Одним из часто используемых методов описания алгоритмов - моделирование посредством абстрактных автоматов [7].

Абстрактные автоматы - вычислительные машины, представленные в виде математических моделей, осуществляющие обработку информации по заданному алгоритму. Проблемами в сфере абстрактных автоматов призвана заниматься теория автоматов – раздел дискретной математики, изучающий способы преобразования информации.

1.4.1 Задачи теории автоматов

Задача анализа. По заданному автомату описать его поведение, установить свойства автомата.

Задача синтеза. Построить автомат с наперед заданным поведением (алгоритмом функционирования). Задачу синтеза принято рассматривать двояко: абстрактный синтез как построение математической модели автомата и структурный синтез как разработку функциональной логической схемы автомата.

Задача полноты. Пусть M — некоторое множество автоматов. Определить, обладает ли совокупность автоматов, составляющих подмножество M' множества M , свойством полноты. Иными словами, если ко всем автоматам подмножества M' конечное число раз применить операцию

суперпозиции, совпадут ли M' и M .

Задача минимизации. Построить автомат, минимальный заданному. Минимальный автомат обладает наименьшим числом компонентов модели (в частности, минимальной мощностью множества так называемых состояний) и при этом функционально эквивалентен заданному автомату.

Задача эквивалентных преобразований. Определить полную систему правил, позволяющую преобразовывать произвольный автомат в любой эквивалентный ему автомат. Частным случаем данной задачи является переход от одной модели автомата к другой. Два автомата функционально эквивалентны, если их поведение одинаково при воздействии одних и тех же последовательностей входных сигналов. В таком случае говорят о совпадении моделей поведения двух автоматов.

1.4.2 Классификация автоматов

Введем понятие конечного автомата. Конечный автомат — это некоторая абстрактная модель, содержащая конечное число состояний чего-либо. Используется для представления и управления потоком выполнения каких-либо команд.

Конечные автоматы классифицируются по нескольким параметрам.

По степени определенности. Автоматы делятся на детерминированные и недетерминированные. Отличие между ними заключается в том, что в один момент времени детерминированные автоматы могут находиться только в одном состоянии, в то время как недетерминированные автоматы могут находиться в нескольких состояниях одновременно. Исходя из этого отличия детерминированный автомат можно описать как последовательность из пяти элементов, представленных в формуле 1.

$$M = (Q, \Sigma, \beta, S_0, F), \quad (1)$$

где Q – множество состояний автомата;

Σ – алфавит языка;

β – функция перехода такая, что $\beta: Q \times \Sigma \rightarrow Q$;

$S_0 \in Q$ – начальное состояние;

$F \subseteq Q$ – множество конечных состояний.

Недетерминированный автомат описывается похожим образом – набором элементов, представленным в формуле 2.

$$M = (Q, \Sigma, \Delta, S, F), \quad (2)$$

где Q – множество состояний автомата;

Σ - алфавит языка;

Δ - отношение перехода;

$S \subseteq Q$ – множество начальных состояний;

$F \subseteq Q$ – множество конечных состояний.

По типу. Различают два типа автоматов. Автомат Мили и автомат Мура. Различия между ними заключается в том, что выходной сигнал в автомате Мили зависит от входного сигнала и от предыдущего состояния автомата, тогда как выходной сигнал в автомате Мура зависит только от текущего состояния автомата.

1.4.3 Способы описания вычислительных систем

Описать какую-либо вычислительную системы можно с помощью конечных автоматов. Чтобы описать систему существует несколько способов.

Описание с помощью таблиц переходов и выходов. При табличном способе задания и описания автомата Мили строятся две таблицы – переходов и выходов. В таблице переходов указывается из какого состояния под действием какого сигнала автомат переходит в то или иное состояние. В таблице выходов указывается формируемый выходной сигнал, выдаваемый автоматом в каком-либо состоянии при поступлении какого-либо входного сигнала. В таблице 1 представлено описание работы RS-триггера, с помощью автомата Мили, где S_0 – состояние, при котором триггер закрыт, S_1 –

состояние, при котором триггер открыт, Z_0 – сигнал, подающийся на вход R, Z_1 – сигнал, подающийся на вход S, W_0 – выходной сигнал с инверсного выхода $\sim Q$, W_1 – выходной сигнал с выхода Q.

Таблица 1 - Описание RS-триггера с помощью автомата Мили

Состояние	Входные сигналы	
	Z_0	Z_1
S_0	S_0/W_0	S_1/W_1
S_1	S_0/W_0	S_1/W_1

Автомат Мура описывается с помощью одной отмеченной таблицы, в которой одновременно в столбце с начальным состоянием также указывается и выходной сигнал, вырабатываемый при переходе в это состояние. На пересечении строки и столбца указывается состояние, в которое переходит автомат из данного состояния под действием данного входного сигнала.

В таблице 2 представлен результат описание RS-триггера с помощью автомата Мура, где S_0 – состояние, при котором триггер закрыт, S_1 – состояние, при котором триггер открыт, Z_0 – сигнал, подающийся на вход R, Z_1 – сигнал, подающийся на вход S, W_0 – выходной сигнал с инверсного выхода $\sim Q$, W_1 – выходной сигнал с выхода Q.

Таблица 2 - Описание RS-триггера с помощью автомата Мура

Состояние	Выходной сигнал	Входные сигналы	
		Z_0	Z_1
S_0	W_0	S_0	S_1
S_1	W_1	S_0	S_1

Описание посредством графа. При таком способе автомат задается в виде математического графа. Граф – абстрактный математический объект, представляющий собой множество вершин и набор ребер – соединений между вершинами.

При описании алгоритма с помощью абстрактного автомата строится граф, множеством вершин которого является множество состояний автомата, а множество ребер – множество переходов между состояниями. При описании

автомата Мили в начале дуги, обозначающей переход, записывается входной сигнал, инициирующий данный переход, в конце дуги записывается выходной сигнал.

На рисунке 12 изображен граф автомата Мили, описывающего работу RS-триггера, где S_0 – состояние, при котором триггер закрыт, S_1 – состояние, при котором триггер открыт, Z_0 – сигнал, подающийся на вход R, Z_1 – сигнал, подающийся на вход S, W_0 – выходной сигнал с инверсного выхода $\sim Q$, W_1 – выходной сигнал с выхода Q.

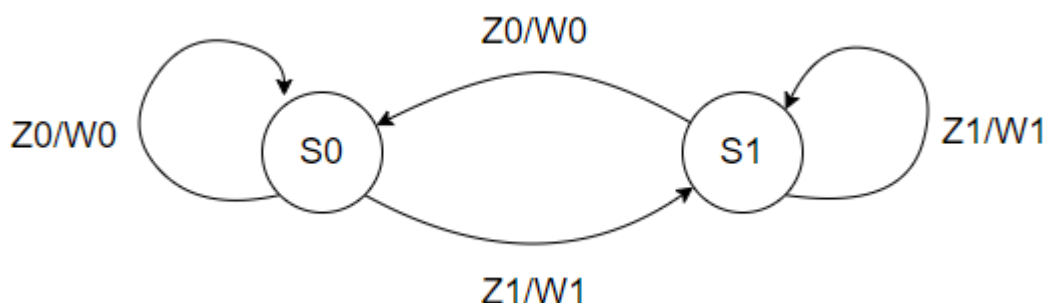


Рисунок 12 - Граф RS-триггера на автомате Мили

Для автомата Мура выходной сигнал записывается рядом с состоянием, в котором он формируется. На рисунке 13 изображен RS-триггер, описанный с помощью автомата Мура, где S_0 – состояние, при котором триггер закрыт, S_1 – состояние, при котором триггер открыт, Z_0 – сигнал, подающийся на вход R, Z_1 – сигнал, подающийся на вход S, W_0 – выходной сигнал с инверсного выхода $\sim Q$, W_1 – выходной сигнал с выхода Q.

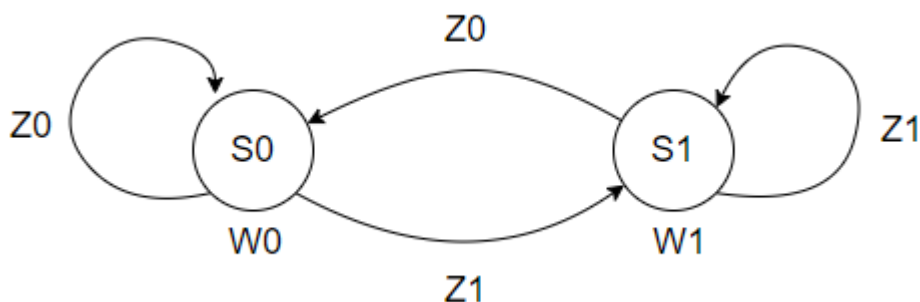


Рисунок 13 - Граф RS-триггера на автомате Мура

1.5 Цели и задачи

Исходя из описанных выше исследований, были сформулированы следующие цели данной работы:

- исследовать методы моделирования распределенных ИС,
- проанализировать способы защиты информации в распределенных ИС.

Также в процессе работы предполагается решить следующие задачи:

- разработка среды моделирования ИС посредством конечных автоматов,
- моделирование в разработанной среде алгоритма защиты данных, используемых в распределенных ИС,
- разработка программы, реализующей исследуемый алгоритм защиты данных в распределенных ИС.

1.6 Анализ существующих методов моделирования

Существуют различные методики описание поведения алгоритмов и систем. Далее приводится сравнение, достоинства и недостатки основных из них.

1.6.1 Цепи Маркова

Представим, что есть набор каких-то событий, связанных друг с другом. Например, первое наступает только после второго, второе — после третьего или четвертого, а третье — после четвертого с вероятностью 30% и так далее. Получается, что каждое новое событие зависит только от предыдущего и не зависит от всех остальных до него [8].

Допустим, есть событие В, которое всегда идет только после события А. При любых других условиях событие В наступит только после того, как произойдет событие А, независимо от каких-то других условий.

Цепи Маркова — это последовательность событий или действий, где каждое новое событие зависит только от предыдущего и не учитывает все остальные события. Такой алгоритм не обладает свойством памяти, т.е. не

учитывает, что было раньше, а анализирует только текущее состояние.

Математически алгоритм цепей Маркова можно записать следующим образом.

$$\mathbb{P}(X_{n+1} = i_{n+1} \mid X_n = i_n, X_{n-1} = i_{n-1}, \dots, X_0 = i_0) = \mathbb{P}(X_{n+1} = i_{n+1} \mid X_n = i_n), \quad (3)$$

где X_n - область значений случайных событий;

$\{I_n\}$ - область значений вероятностей, с которой наступит событие X_n ;

n - количество событий.

Метод моделирование алгоритмов с помощью цепей Маркова не подходит для достижения поставленных целей данной работы, т.к. подразумевает моделирование событий, которые могут наступить с какой-либо вероятностью, что не является характеристикой алгоритма, моделируемого в данной работе.

1.6.2 Сети Петри

Сеть Петри — математический объект, используемый для моделирования динамических дискретных систем, предложенный Карлом Петри в 1962 году.

Определяется как двудольный ориентированный мультиграф, состоящий из вершин двух типов — позиций и переходов, соединенных между собой дугами. Вершины одного типа не могут быть соединены непосредственно. В позициях могут размещаться метки (маркеры), способные перемещаться по сети. Событием называют срабатывание перехода, при котором метки из входных позиций этого перехода перемещаются в выходные позиции. События происходят мгновенно либо одновременно, при выполнении некоторых условий [9].

Сеть Петри есть мультиграф, так как он допускает существование кратных дуг от одной вершины графа к другой. Так как дуги являются направленными, то это ориентированный мультиграф. Вершины графа можно разделить на два множества (позиции и переходы) таким образом, что каждая дуга будет направлена от элемента одного множества (позиций или переходов)

к элементу другого множества (переходов или позиций); следовательно, такой граф является двудольным ориентированным мультиграфом. Пример сети Петри приведен на рисунке 14, где P - множество позиций сети, T - множество переходов.

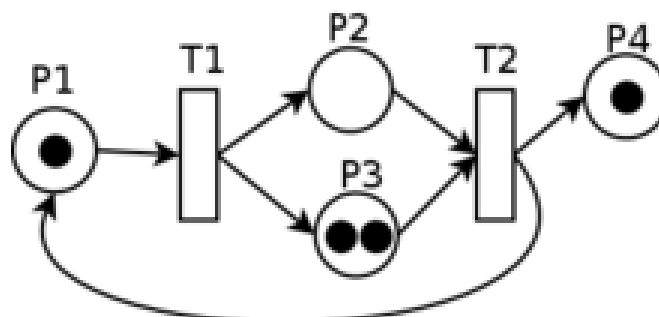


Рисунок 14 - Модель сети Петри

Изначально сети Петри разрабатывались для моделирования систем с параллельными взаимодействующими компонентами, что также не удовлетворяет требованиям моделирования исследуемого алгоритма защиты данных.

1.6.3 Конечные автоматы

Конечный автомат (КА) — математическая абстракция, модель дискретного устройства, имеющего один вход, один выход и в каждый момент времени находящегося в одном состоянии из множества возможных. Является частным случаем абстрактного дискретного автомата, число возможных внутренних состояний которого конечно [10].

При работе на вход КА поступают последовательно входные воздействия, а на выходе КА формирует выходные сигналы. Обычно под входными воздействиями принимают подачу на вход автомата символов одного алфавита, а на выход КА в процессе работы выдает символы в общем случае другого, возможно даже не пересекающегося со входным, алфавита [11]. Пример конечного автомата представлен на рисунке 15.

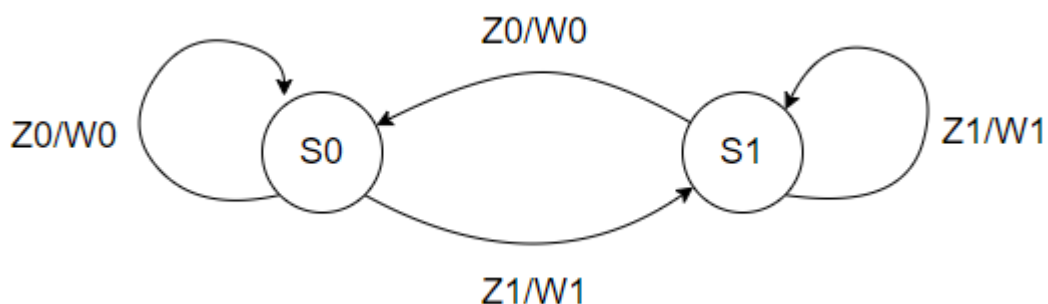


Рисунок 15 - Конечный автомат

Примерами физической реализации КА могут служить любые цифровые системы, например, компьютеры или некоторые логические узлы компьютеров с памятью — триггеры и другие устройства [11]. Благодаря перечисленным свойствам конечные автоматы - наиболее подходящий математический объект для моделирования исследуемого алгоритма. В качестве метода моделирования был выбран именно конечный автомат.

1.7 Выбор языка программирования

При выборе языка программирования необходимо учитывать модульность языка - способность четко разграничивать приложения и библиотеки на отдельные модули, а также просто и эффективно использовать эти модули вместе. Исходя из этого качества для такого языка программирования есть немало сторонних библиотек и подключаемых модулей, что способствует широкой области применения языка.

Также важно учитывать, чтобы язык был объектно-ориентированный. Объектно-ориентированные языки предоставляют разработчику возможность правильно спроектировать структуру приложения для удобной и корректной разработки конечного продукта [6]. Также немаловажными факторами разработки является чистота кода и производительность [7]. В настоящее время наиболее распространенными языками для разработки оконных приложения являются C++, Java, JavaScript, C#.

1.7.1 Язык программирования C++

C++ - объектно-ориентированный, компилируемый, строго типизированный язык программирования. Этот язык удовлетворяет основным

парадигмам объектной ориентированности, таким как полиморфизм, наследование, инкапсуляция. К недостаткам языка можно отнести несовершенство механизма работы с памятью, и сложный синтаксис языка.

1.7.2 Язык программирования JavaScript

JavaScript – скриптовый интерпретируемый язык, предназначенный для исполнения в среде браузера. В основном применяется в разработки Web-страничек. JavaScript поддерживает императивный, декларативный и объектно-ориентированный стили программирования. Так как JavaScript интерпретируемый язык безопасность и производительность напрямую зависит от конкретной реализации интерпретатора.

1.7.3 Язык программирования C#

C# – объектно-ориентированный, компилируемый, строго типизированный язык программирования [12]. Является производным от языка C++. В этом языке хорошо развита поддержка оконных приложений. Также C# является объектно-ориентированным языком, в виду чего ему тоже свойственны такие парадигмы как полиморфизм, наследование и инкапсуляция. C# обладает хорошей производительностью и безопасностью. В качестве языка для разработки среды моделирования был выбран именно он.

1.7.4 Язык программирования Java

Java – кроссплатформенный строго типизированный, объектно-ориентированный язык программирования. Кроссплатформенность данного языка достигается благодаря наличию нем такого средства, как jvm – java virtual machine. При компиляции исходный код, компилируется в так-называемый байт-код, способный исполняться виртуальной машиной на практически любой физической машине. Для Java разработано большое количество подключаемых библиотек и сервисов, в том числе и в области защиты информации. В качестве языка для разработки приложения, реализующего исследуемый алгоритм защиты информации был выбран именно он.

1.8 Выводы

Основываясь на результатах проведенного анализа, было принято решение разработать собственную среду проектирования и моделирования информационных систем, которая будет решать такие задачи, как проектирование сложных вычислительных и информационных систем, анализ систем с целью выявления узких мест и уязвимостей еще на стадии проектирования.

Также было принято решение промоделировать в разработанной среде алгоритм защиты персональных данных пользователей в распределенных информационных системах, а также разработать приложение, реализующее исследуемый алгоритм.

В качестве языка разработки среды моделирования был выбран язык C# т.к. одним из важных критериев разработанной среды является удобный и понятный для пользователя интерфейс. Такой язык программирования, как C# позволяет удовлетворить данную потребность.

В качестве языка разработки приложения был выбран язык Java, т.к. он является объектно-ориентированным и имеет большое количество подключаемых библиотек и сервисов, необходимых для разработки приложения.

2 Моделирование

В качестве решения вышеупомянутой проблемы была разработана и отлажена среда моделирования и симуляции вычислительных систем посредством парсинга системы канонических уравнений (СКУ). Принципиальное отличие разработанной среды проектирования вычислительных систем заключается в том, что с помощью данной среды пользователь может спроектировать и промоделировать поведение распределенных и высоконагруженных информационных систем.

2.1 Описание работы разработанной среды

Моделирование предполагается проводить с помощью построения конечного автомата, описывающего поведение системы. Для удовлетворения своих потребностей пользователю достаточно описать поведение исходной системы аналитическим образом, то есть с помощью системы канонических уравнений. СКУ задаются как набор формул в нормальной дизъюнктивной форме (ДНФ). В качестве операндов самого уравнения выступают конъюнкты, состоящие из предыдущего состояния и выходного сигнала предыдущего состояния, само же уравнение приравнивается к новому состоянию. Количество конъюнктов определяет количество вершин графа автомата, из которых осуществляется переход.

После ввода СКУ среда приступает к анализу введенных данных. Происходит построчное считывание информации из текстового пространства и результат считывания передается разработанному парсеру СКУ. Данный алгоритм анализирует переданную информации с помощью механизма регулярных выражений и создает множества, описывающие компоненты графа.

После того как все данные из текстового пространства прочитаны и проанализированы, исходя из сформировавшихся объектов строится модель эмулируемой системы, представленная в виде конечного автомата. На графическом пространстве поочередно прорисовывается каждый компонент

графа, созданный парсером СКУ. После отрисовки элементов графа управление возвращается пользователю.

Пользователь имеет возможность взаимодействовать с графическими элементами, перемещать, выделять и устанавливать их в сигнальное состояние. После совершения каких-либо действий пользователя, поочередно анализируются множества объектов, и, в случае необходимости, происходит перерисовка компонентов, потерпевших изменение, после чего управление возвращается пользователю.

Если пользователь начал симуляцию, на каждом ее шаге определяются состояния компонентов графа, если какой-то элемент потерпел изменение, происходит перерисовка графа, и затем управление возвращается пользователю. На рисунках 16 и 17 представлены UML-диаграммы последовательности и вариантов использования разработанной среды.

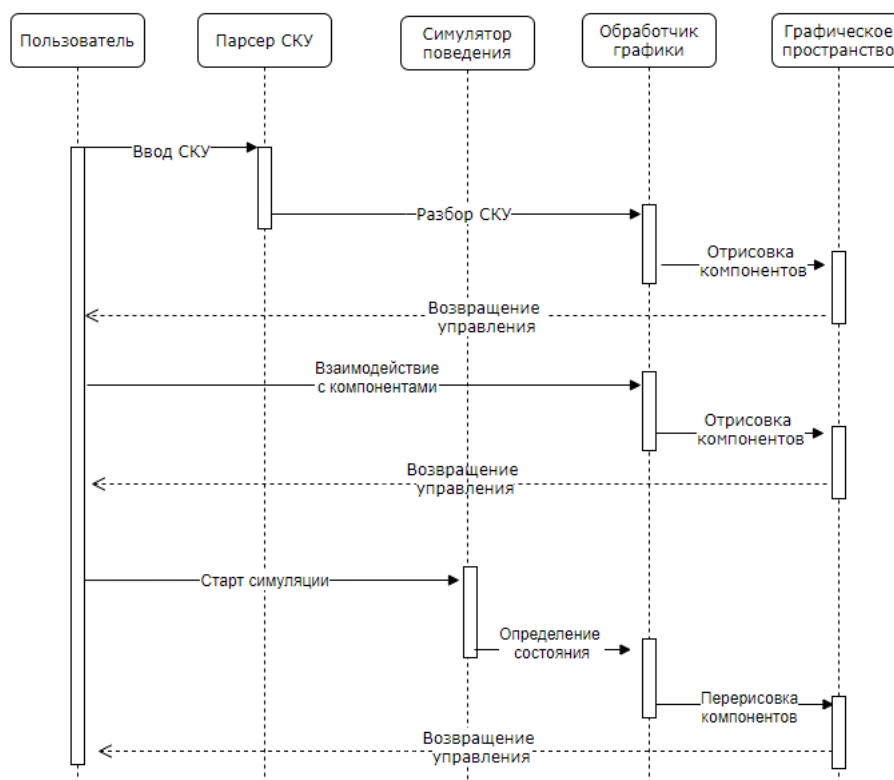


Рисунок 16 - UML-диаграмма последовательности работы алгоритма

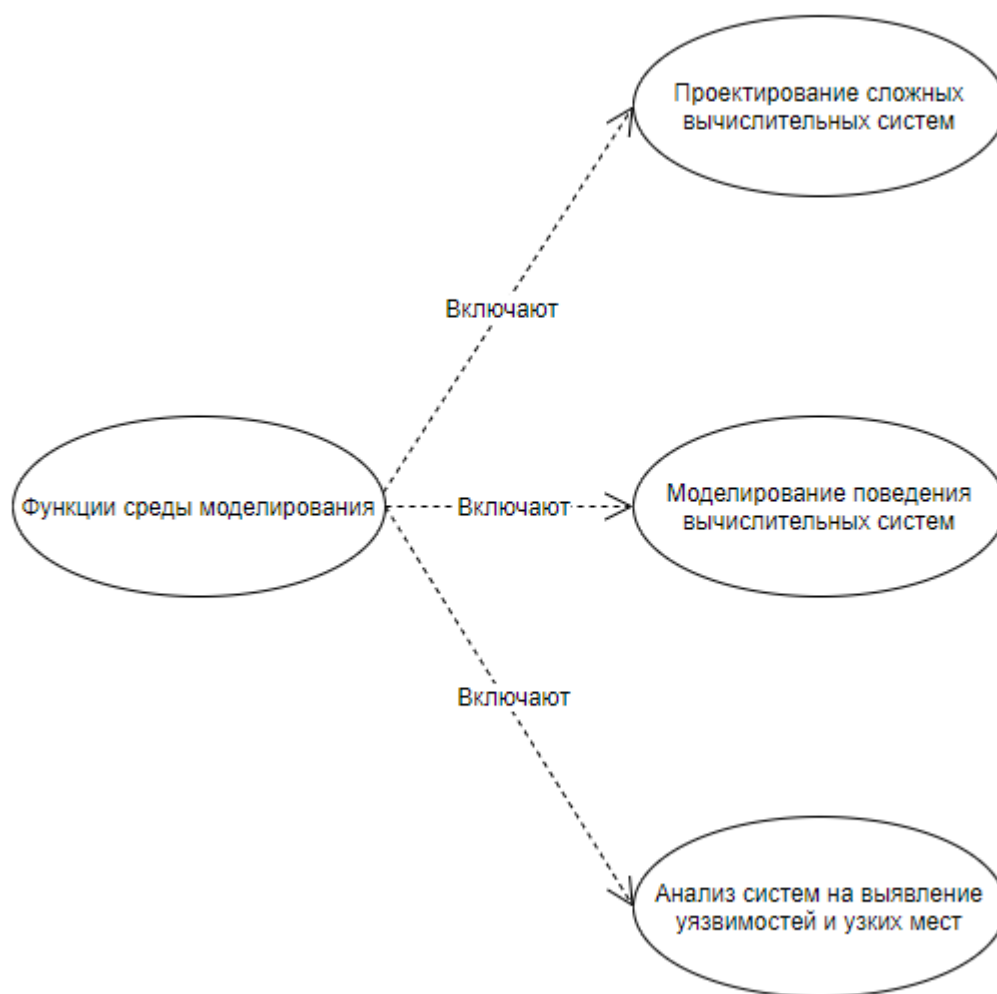


Рисунок 17 - UML-диаграмма вариантов использования функций среды моделирования

2.2 Описание алгоритмов

В ходе выполнения работы, были разработаны и отлажены следующие алгоритмы.

2.2.1 Алгоритм разбора и анализа СКУ

Алгоритм производит построчное считывание данных из текстового пространства, предназначенного для ввода СКУ. Каждая строка анализируется с помощью механизма обработки регулярных выражений. В среде задается шаблон, согласно которому происходит разбор входной строки. Шаблон разбивается на 3 группы.

Первая группа производит анализ строки с целью нахождения конечного состояния, характерного для анализируемой строки.

Вторая группа отвечает за поиск начальных состояний, из которых можно перейти в конечное состояние.

Третья группа обрабатывает строку с целью поиска входных сигналов, под действием которых происходит переход.

UML-диаграмма деятельности изображена на рисунке 18.

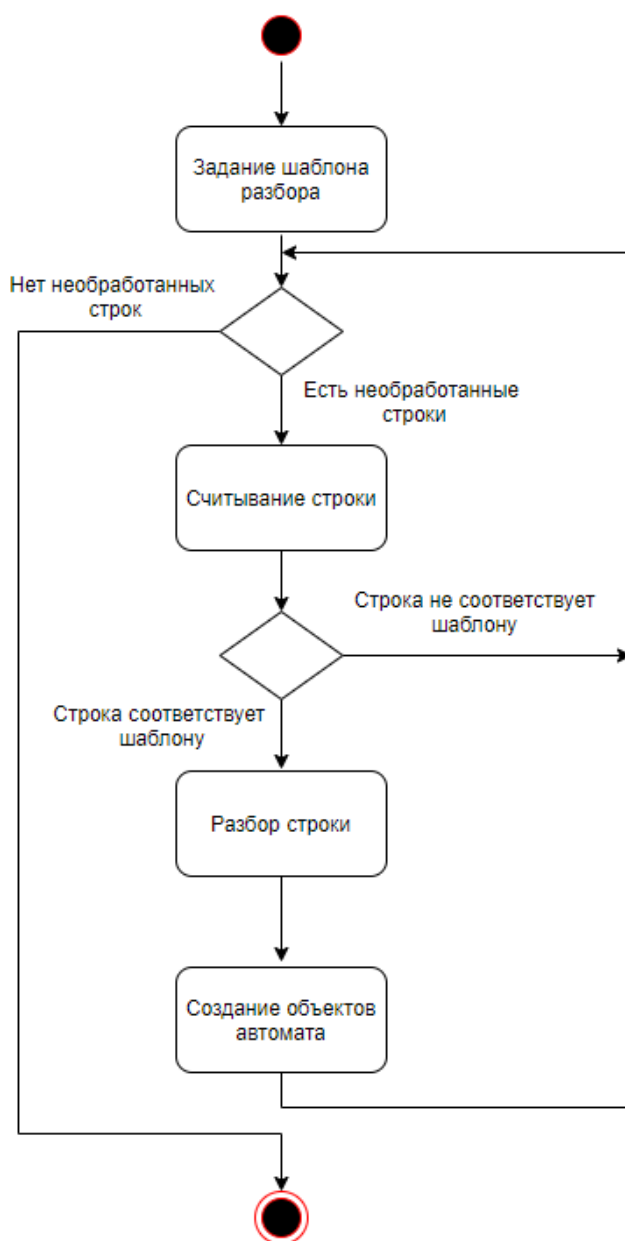


Рисунок 18 - UML-диаграмма деятельности разбора СКУ

2.2.2 Алгоритм отрисовки компонентов

После того, как проанализированы все строки СКУ, происходит отображение созданных объектов на графическом пространстве. Перед отрисовкой у каждого компонента рассчитываются координаты точек, в которых происходит отображение данного элемента. При наведении курсора мыши на тот или иной элемент, происходит его подсвечивание. Данный эффект достигается за счет следующего алгоритма.

Шаг1. Всякий раз при движении мыши по графическому пространству, в системе устанавливается и обрабатывается соответствующее событие (event).

Шаг 2. Среди свойств события присутствуют такие свойства, как координата текущего положения мыши по оси x , и координата текущего положения мыши по оси y . Точка начала координат расположена в крайней левой верхней точки графического пространства, т.е. значение каждой координаты означает количество точек от точки начала координат до точки текущего расположения мыши по соответствующей оси.

Шаг 3. После того, как в среде инициализировались координаты текущего положения мыши, поочередно происходит анализ всех графических компонентов графа.

Шаг 4. Как описывалось ранее, у каждого элемента графа есть координаты его вершин. Исходя из значений координат этих вершин можно вычислить площадь графического пространства, занимаемого тем или иным элементом, а также крайние точки этой области. Поочередно, у каждого графического компонента вычисляется занимаемая площадь и крайние точки этой площади.

Шаг 5. После того, как все необходимые расчеты были проведены, данные о расположении каждого элемента на графическом пространстве сравниваются с координатами текущего положения курсора мыши.

Шаг 6. Если курсор мыши находится в пределах области графического пространства, занимаемого тем или иным элементом, данный элемент устанавливается в специальное состояние.

Шаг 7. Если в процессе обработки события движения мыши какие-то флаги элементов графа поменяли свое значение, например какой-то элемент был не активен, а стал активен и наоборот, происходит перерисовка изменившихся компонентов, в соответствии с нужными стилями.

UML-диаграмма деятельности приведена на рисунке 19.

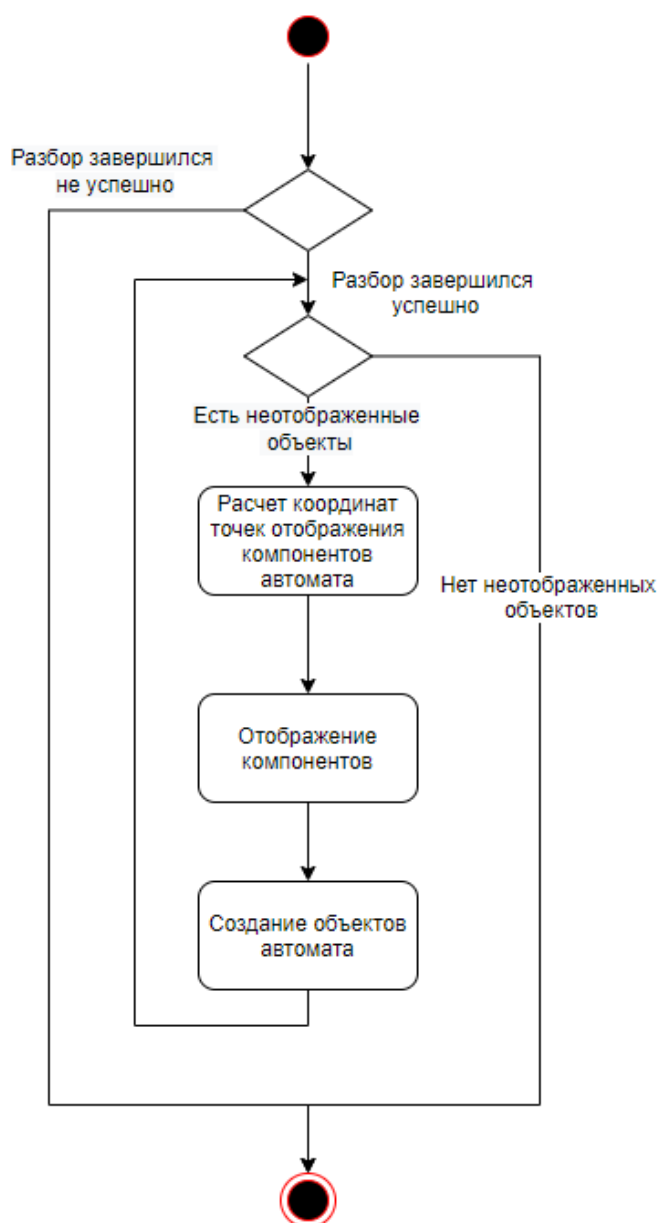


Рисунок 19 – UML-диаграмма деятельности алгоритма отрисовки

2.2.3 Алгоритм моделирования поведения системы

После того, как все инициализирующие операции были завершены, происходит симуляция работы вычислительной системы [13].

Алгоритм проверяет, указано ли какое-либо начальное состояние. Если начальное состояние не указано, будет выведено сообщение об ошибке.

Если начальное состояние установлено алгоритм приступает к моделированию поведения. Исходя из информации о активности входных сигналов алгоритм на каждом шаге определяет состояние в какое переходит моделируемая система, либо остается в исходном состоянии если никакой переход еще не сработал [14]. Цикл продолжается до тех пор, симуляция не остановлена.

Также алгоритм предусматривает возможность пошагового моделирования поведения. После каждого такта моделирования, управление возвращается пользователю для анализа следующего действия.

Цикл повторяется до тех пор, пока симуляция не остановлена [15]. UML-диаграмма деятельности приведена на рисунке 20.

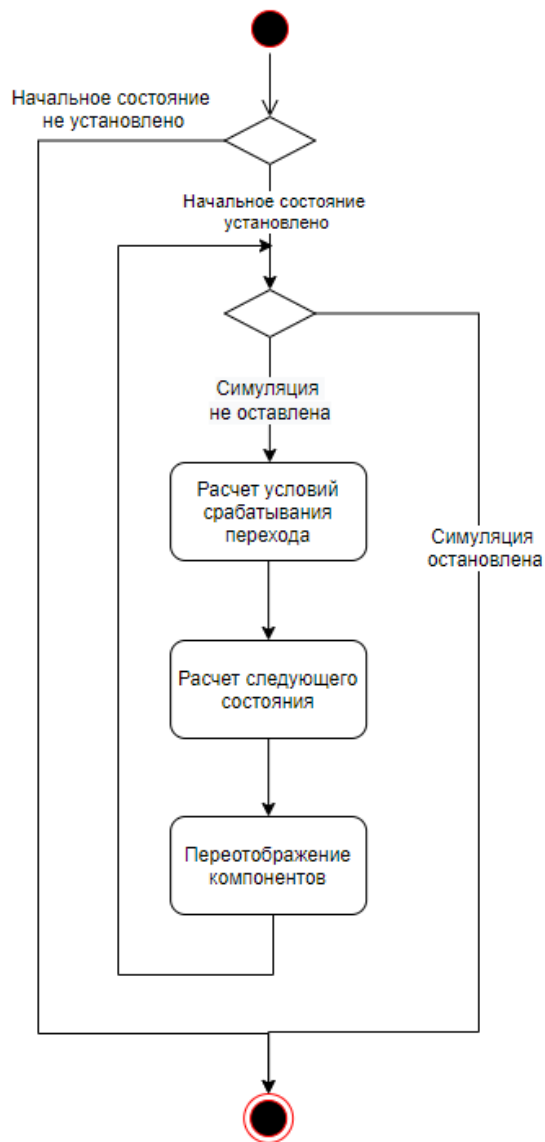


Рисунок 20 - UML-диаграмма деятельности алгоритма симуляции

2.3 Моделирование алгоритма OAuth2

Для решения поставленных задач с помощью разработанной среды моделирования был смоделирован алгоритм работы протокола OAuth2.0. В частности, разработан конечный автомат, описывающий работу данного алгоритма. В качестве типа КА был выбран автомат Мили. Характерная особенность автомата Мура в том, что выходной сигнал в каждом состоянии КА зависит только от этого состояния. Такой тип КА наиболее предпочтительный и удобный для описания данного алгоритма.

2.3.1 Задание КА методом СКУ

Конечный автомат можно задать с помощью системы канонических уравнений (СКУ), описывающей все реализуемые в КА частные события, состояния и переходы в них.

В текстовой форме конечный автомат можно задать следующим образом.

Шаг 1. В начальном состоянии S_0 автомат предлагает пользователю перейти на сервер авторизации.

Шаг 2. Если пользователь еще не выполнил вход в свою учетную запись, на вход автомата поступает входной сигнал x_1 , и КА переходит в состояние S_1 , где пользователь должен авторизоваться на форме авторизации. На выходе КА формируется сигнал w_0 .

Шаг 3. Если авторизация прошла успешно, на вход автомата поступает входной сигнал x_0 , под действием которого КА переходит в состояние S_2 , где сервер перенаправляет пользователя на форму, где тот принимает решение, выдавать ли клиентскому приложению права доступа к своему ресурсу на сервере ресурсов или нет. На выходе КА формируется сигнал w_0 .

Шаг 4. Если права выданы пользователем, на вход автомата поступает входной сигнал x_2 , под действием которого КА переходит в состояние S_3 и запрашивает код авторизации у сервера авторизации. На выходе КА формируется сигнал w_0 . Иначе КА переходит в ошибочное состояние Se с выходным сигналом w_1 и отображает соответствующее сообщение.

Шаг 5. Если код авторизации успешно получен, на вход автомата поступает входной сигнал x_3 , под действием которого КА переходит в состояние S_5 , где происходит обмен кода на токен авторизации. На выходе КА формируется сигнал w_0 . Иначе КА переходит в ошибочное состояние Se с выходным сигналом w_1 и отображает соответствующее сообщение.

Шаг 6. Если токен успешно получен, на вход автомата поступает входной сигнал x_4 , под действием которого КА переходит в конечное состояние S_k . На выходе КА формируется сигнал w_0 . Иначе КА переходит в

ошибочное состояние Se с выходным сигналом $w1$ и отображает соответствующее сообщение.

В таком случае, СКУ конечного автомата будет выглядеть согласно формулам.

$$S0 = Se \& X5;$$

$$S1/w0 = S0 \& X1;$$

$$S2/w0 = (S0|S1) \& X0;$$

$$S3/w0 = S2 \& X2;$$

$$S4/w0 = S3 \& X3;$$

$$Sk/w0 = S4 \& X4;$$

$$Se/w1 = (S2|S3|S4) \& Xe.$$

2.3.2 Задание КА в виде графа

Аналогично, КА, описывающий алгоритм протокола можно задать в графической форме, т.е. с помощью графа. Граф автомата, приведен на рисунке 21.

На рисунке применяются следующие обозначения:

- начальное состояние $s0$,
- состояние авторизации пользователя на сервере авторизации $s1$,
- состояние предоставления пользователем прав к ресурсу $s2$,
- состояние запроса кода авторизации $s3$,
- состояние обмена кода авторизации на токен авторизации $s4$,
- конечное состояние, когда токен получен sk ,
- ошибочное состояние se ,
- сигнал перехода в окно выдачи прав $x0$,
- сигнал перехода в окно авторизации $x1$,
- сигнал предоставления пользователем прав к своему ресурсу на сервере ресурсов $x2$,

- сигнал получения кода авторизации $x3$,
- сигнал получения токена $x4$,
- сигнал возврата в начальное состояние $x5$,
- сигнал возникновения ошибки x_e ,
- выходной сигнал успешного завершившейся операции $w0$,
- выходной сигнал, ошибочного завершения операции $w1$.

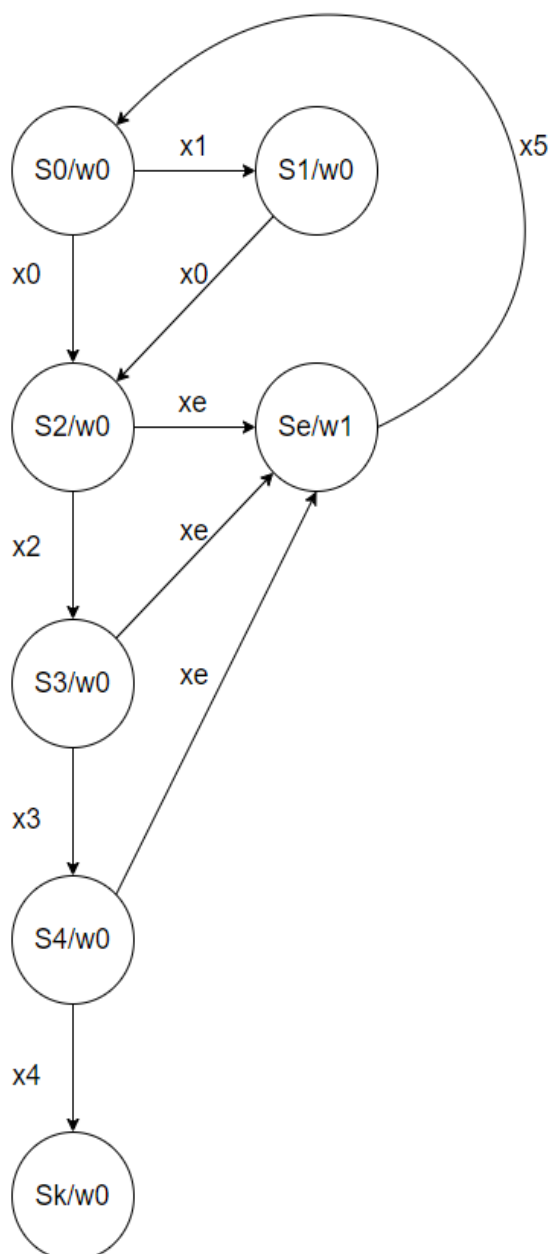


Рисунок 21 - Конечный автомат OAuth2

2.3.3 Задание КА с методом отмеченных таблиц

Также КА возможно задать с помощью таблицы переходов и выходов. Совместим таблицы переходов и выходов в одну таблицу. Строки таблицы означают исходное состояние, столбцы - входные сигналы. На пересечении строки и столбца указывается в какое состояние перейдет КА и какой выходной сигнал формируется на выходе. Для данного КА такая таблица будет выглядеть подобно таблице 3.

Таблица 3 – Отмеченная таблица КА

Состояние	Входные сигналы						
	X ₀	X ₁	X ₂	X ₃	X ₄	X ₅	X _e
S ₀ /w ₀	S ₂	S ₁	-	-	-	-	-
S ₁ /w ₀	S ₂	-	-	-	-	-	-
S ₂ /w ₀	-	-	S ₃	-	-	-	S _e
S ₃ /w ₀	-	-	-	S ₄	-	-	S _e
S ₄ /w ₀	-	-	-	-	S _k	-	S _e
S _e /w ₁	-	-	-	-	-	S ₀	-
S _k /w ₀	-	-	-	-	-	-	-

2.3.4 Моделирование алгоритма в разработанной среде

Промоделируем заданный конечный автомат в разработанной среде моделирования. Одним из основных достоинств среды является возможность построить граф КА на основе системы канонических уравнений (СКУ). В таком случае СКУ вводится в форме для ввода, представленной на рисунке 22.

Первой строкой задается множество входных сигналов, под действием которых КА переходит из одного состояния в другое.

Далее, построчно указываются правила, из какого состояния в какое и под действием каких сигналов переходит КА. Ввод СКУ для данного КА представлен на рисунке 23.

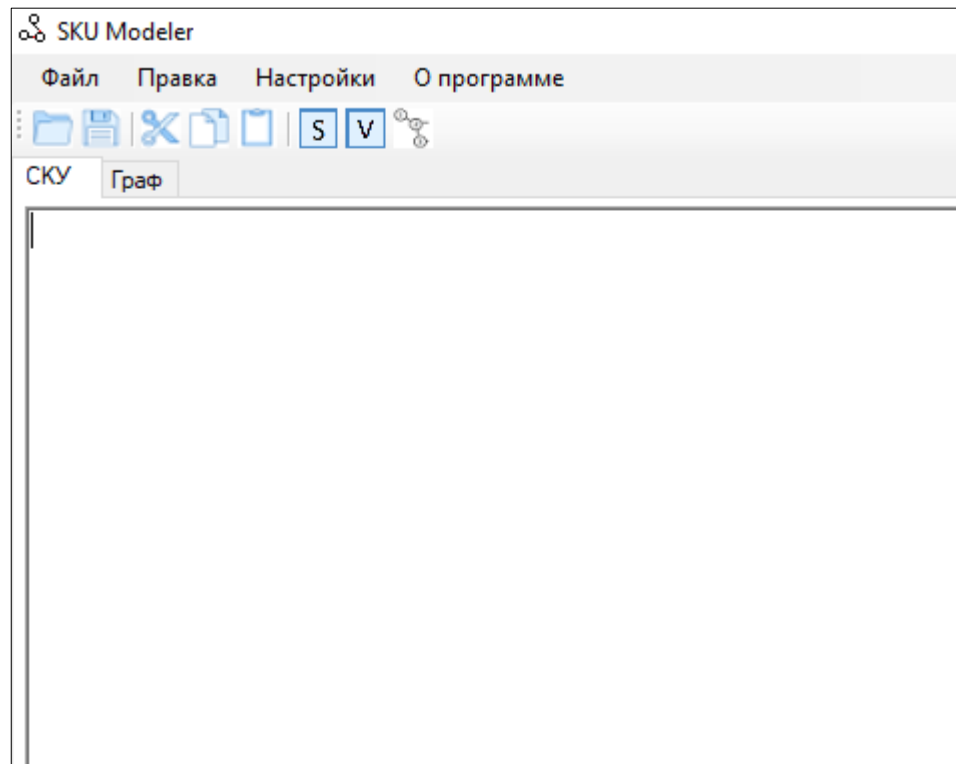


Рисунок 22 - Поле для ввода SKU

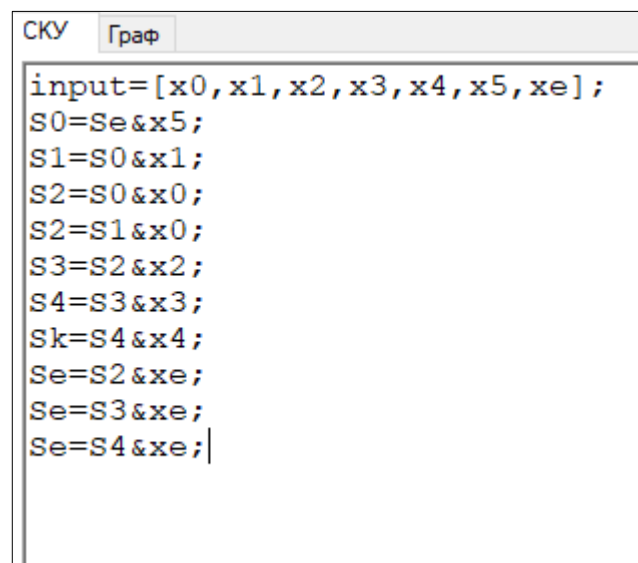


Рисунок 23 - Ввод SKU

После ввода SKU требуется нажать кнопку “Построить граф”, после чего запускается лексический анализ введенной системы, по результатам которого строится граф КА, представленный на рисунке 24.

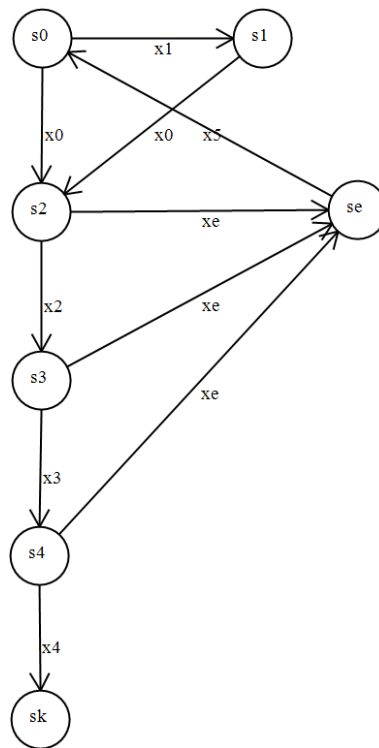


Рисунок 24 - Граф КА

Если в процессе разбора SKU произойдет ошибка, среда уведомит об этом пользователя и граф не построится. Общий интерфейс среды моделирования приведен на рисунке 25.

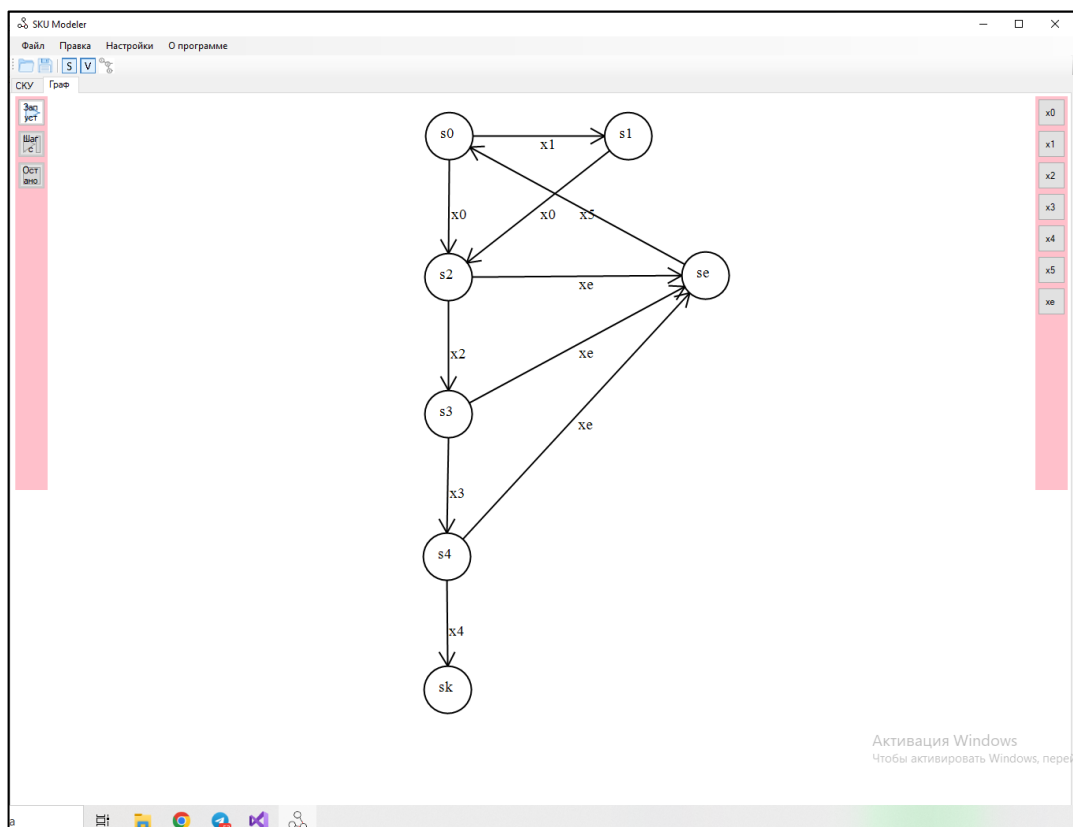


Рисунок 25 - Интерфейс среды моделирования

После того, как разбор СКУ и построение графа завершились успешно, можно приступить непосредственно к симуляции работы КА. Для этого необходимо кликом мыши выделить начальное состояние КА. Результат этого действия приведен на рисунке 26.

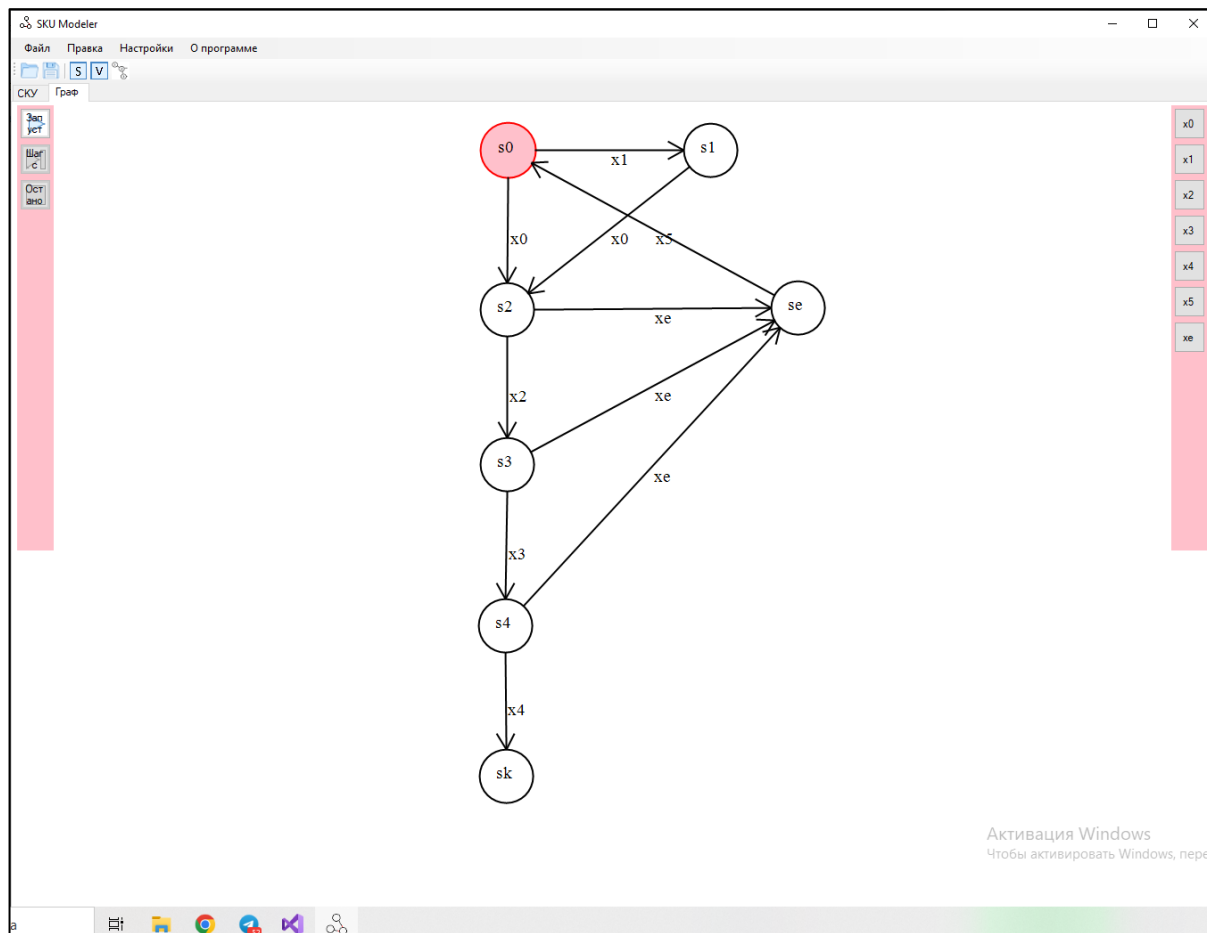


Рисунок 26 - Установка начального состояния

Чтобы начать симуляцию требуется нажать кнопку “Запустить”, расположенную на боковой панели слева.

В среде также предусмотрена возможность проводить симуляцию пошагово, т.е. КА будет менять свое состояние только при нажатии по кнопке “Шаг с остановом”.

Чтобы остановить симуляцию требуется нажать кнопку “Остановить”. Результат приведен на рисунке 27.

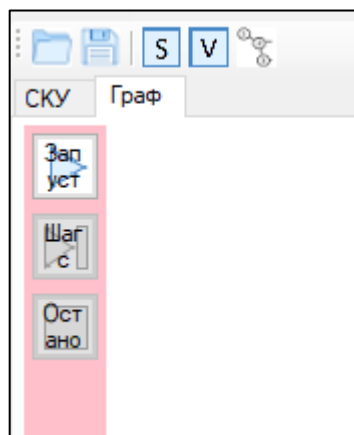


Рисунок 27 - Органы управления

В процессе симуляции входные сигналы можно подавать с помощью панели справа. Для каждого входного сигнала предусмотрена кнопка, при нажатии которой сигнал подается на вход КА. Поданный входной сигнал отображается красным цветом. Панель приведена на рисунке 28.

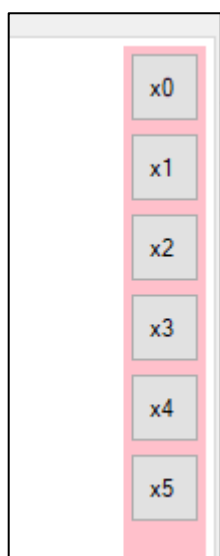


Рисунок 28 - Панель входных символов

Пошаговый процесс симуляции представлен на рисунках 29-35.

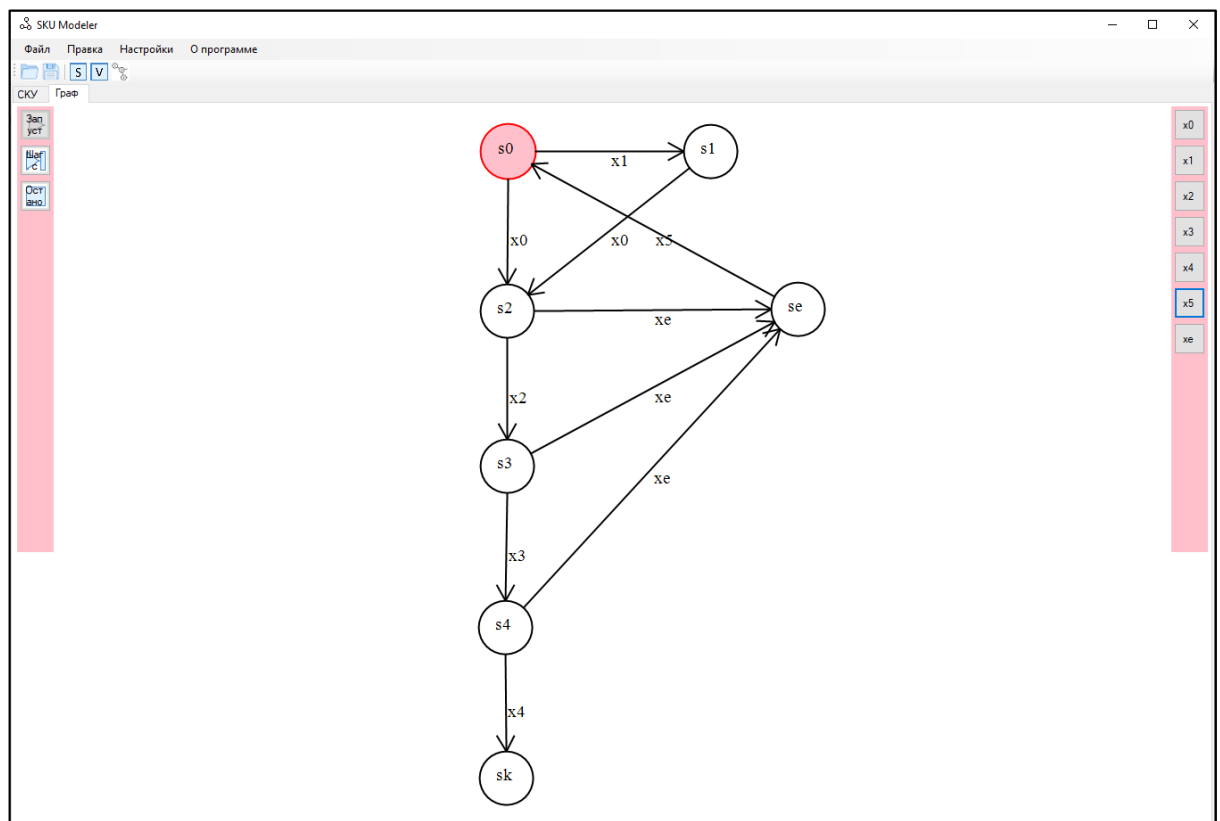


Рисунок 29 – Состояние s0

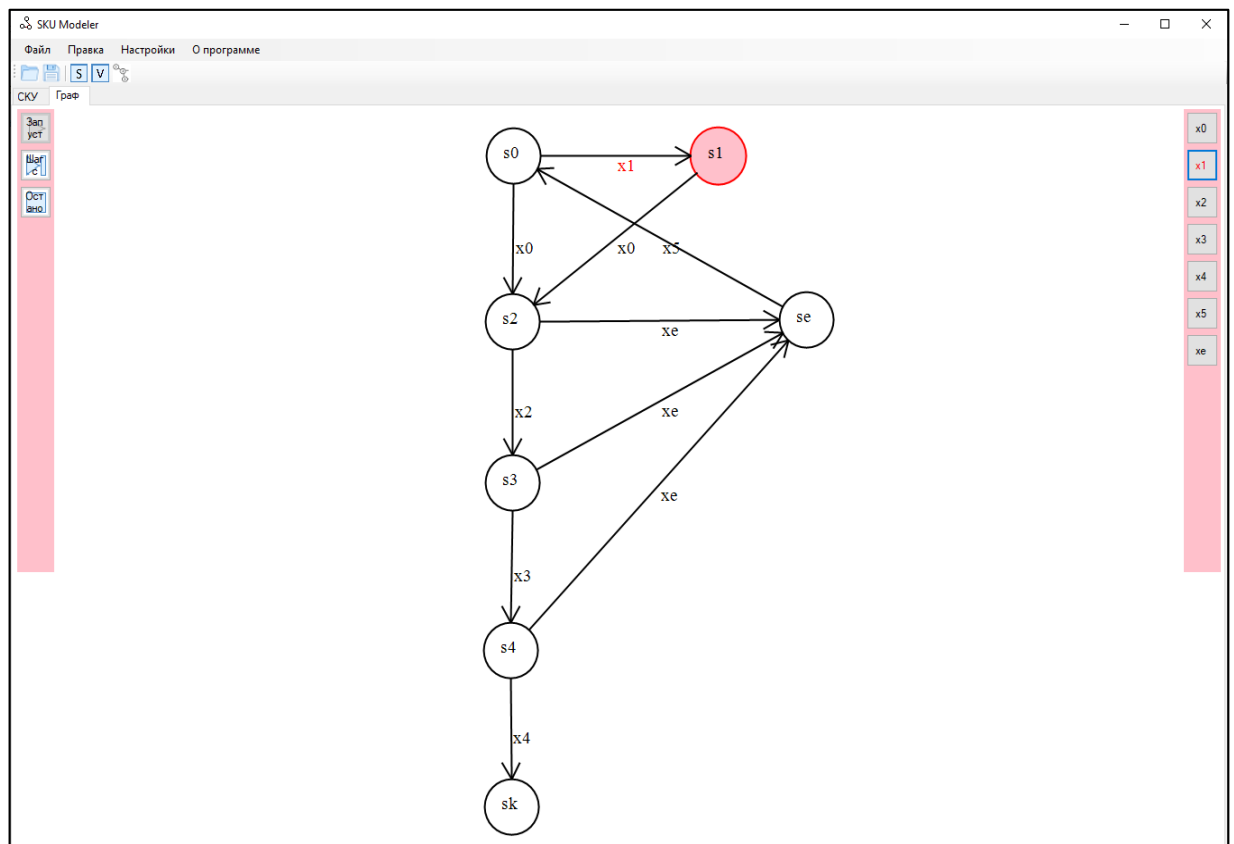


Рисунок 30 - Состояние s1

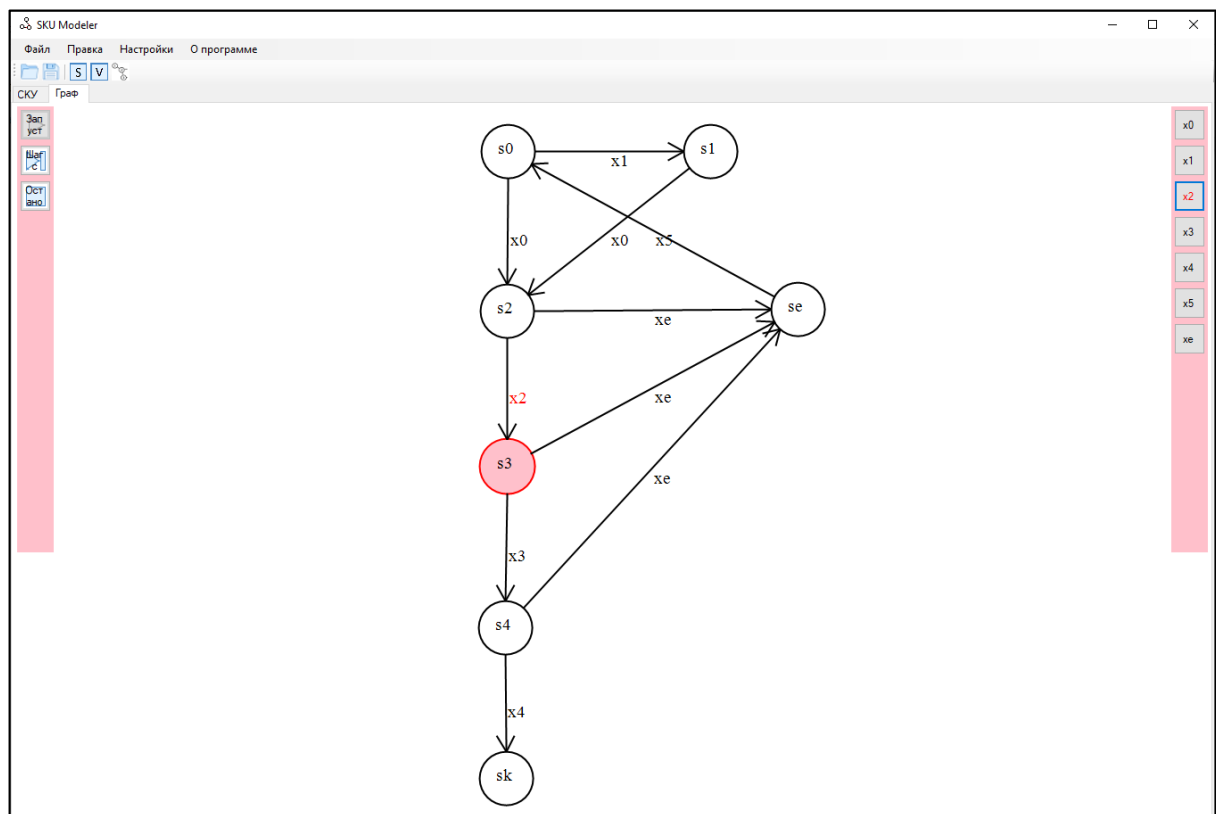


Рисунок 31 - Состояние s3

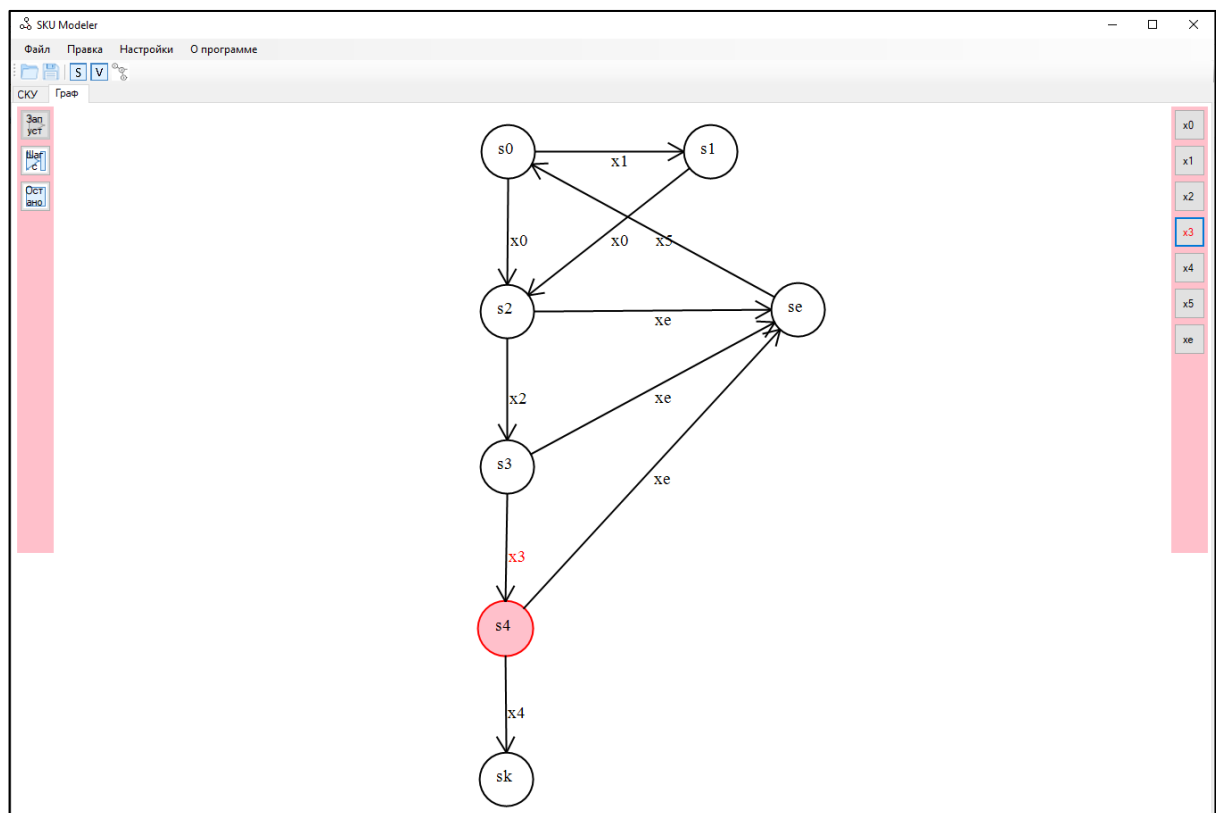


Рисунок 32 - Состояние s4

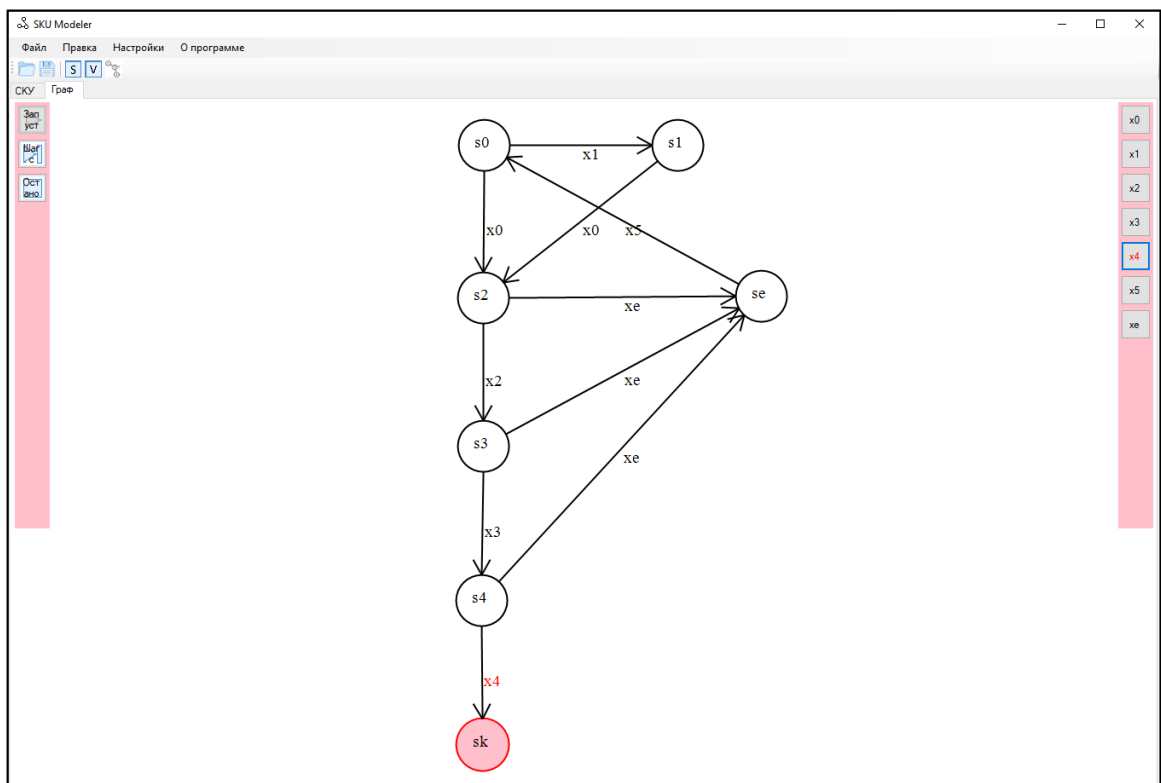


Рисунок 33 – Состояние sk

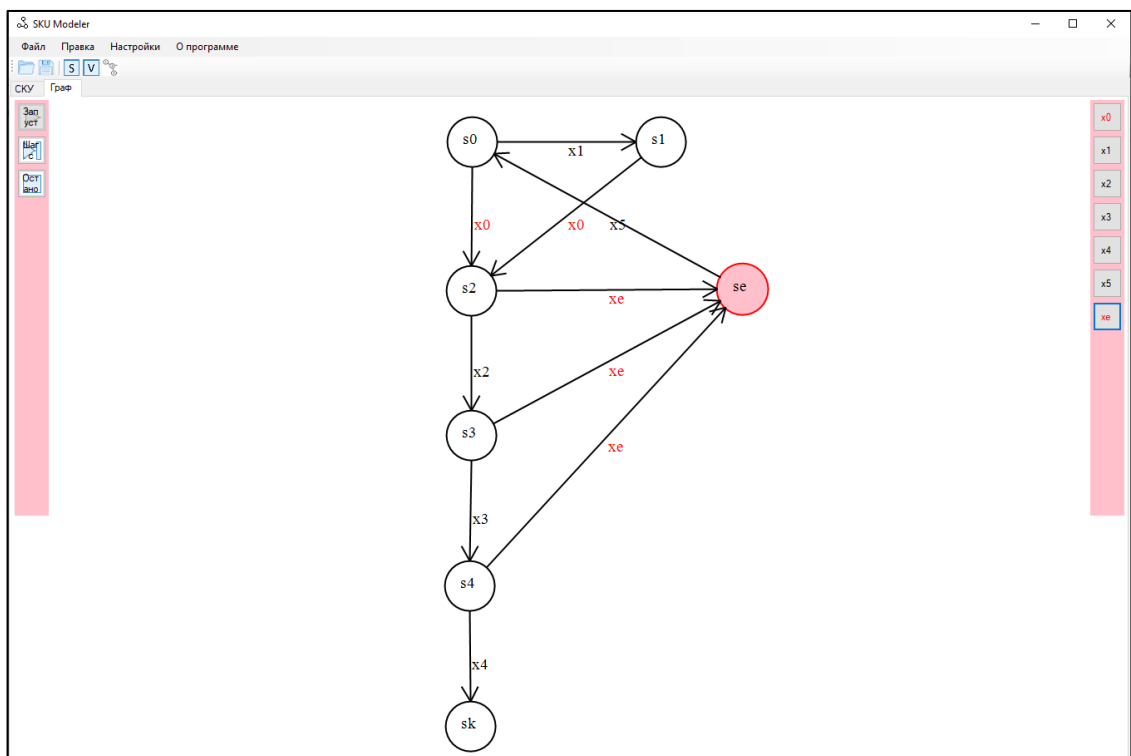


Рисунок 34 - Состояние se

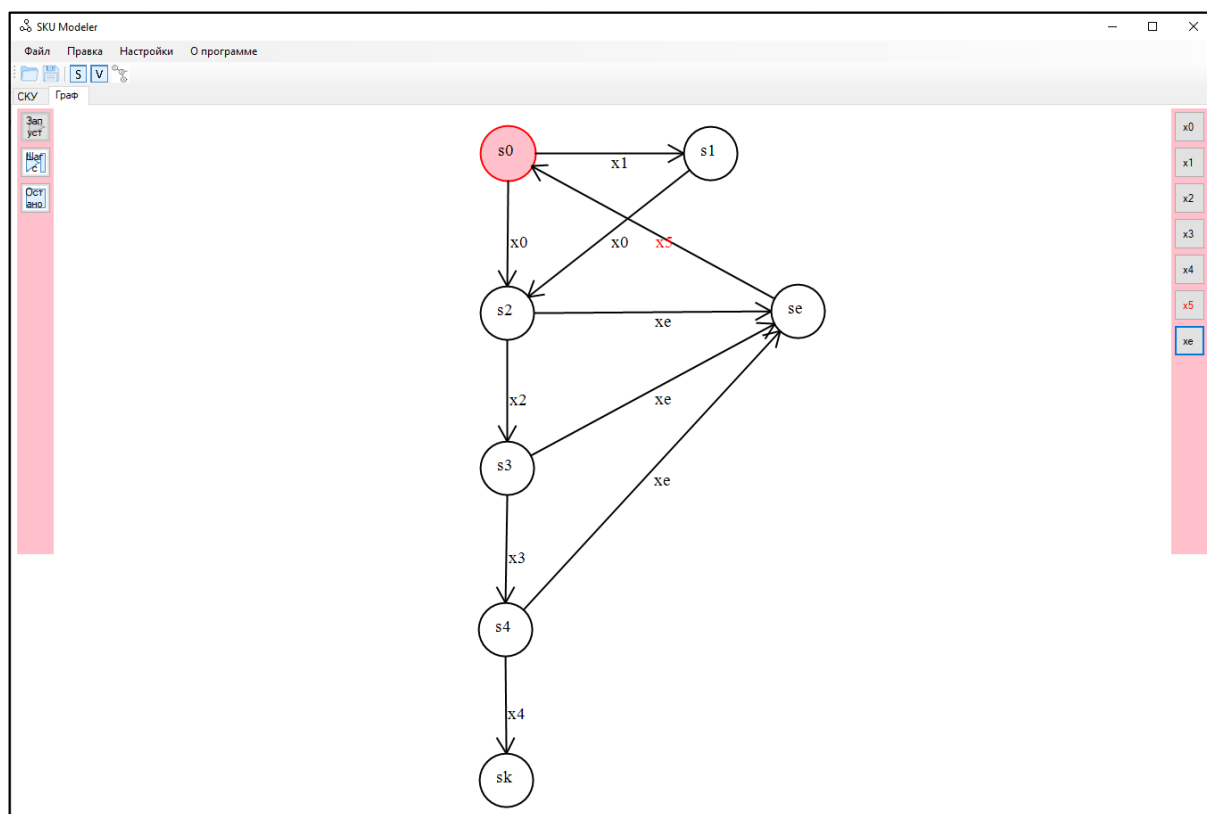


Рисунок 35 – исходное состояние

2.4 Моделирование OIDS

Основная задача протокола OIDS - распознавать, какой конкретно пользователь запрашивает ресурс. Решается эта задача следующим образом. В запросе на получение кода авторизации добавляется дополнительный параметр запроса `openid`, сообщающий серверу, что в ответе нужно вернуть помимо токенов доступа, так называемый `id_token`.

`Id_token` содержит информацию о пользователе в формате `json`, но в закодированном виде. Алгоритм кодировки обычно `Base64Url`. Т.е. данные о пользователе сохраняются в `json` объект, который следом кодируется в строку, которая возвращается клиенту в виде `id_token`. Данный метод передачи информации называется `JWT (Json Web Token)`. Фактически, `jwt`-токен это строка, состоящая из трех частей, разделенных точками.

Первая часть описывает алгоритм кодирования токена и его тип.

Вторая часть содержит непосредственно полезную нагрузку (`payload`), представляющую собой информацию о пользователе.

Третья часть является зашифрованной подписью токена, необходимая для проверки подлинности токена на стороне сервера авторизации. Алгоритм шифрования указан в первой части токена и работает в соответствии с формулой.

Конечный автомат, описывающий протокол OIDC полностью совпадает с протоколом OAuth2, описанном в предыдущем пункте, т.к. протокол полностью строится на основе OAuth2.

2.5 Выводы

В ходе работы была разработана среда моделирования, позволяющая описывать поведение распределенной вычислительной системы, как конечный автомат с его состояниями, переходами, входными и выходными сигналами.

Главным достоинством разработанной среды является возможность проводить симуляцию работы моделируемого КА, подавая на вход и считывая с выхода символы из алфавита КА, а также в режиме реального времени наблюдать в каком состоянии сейчас находится автомат, и какой переход сработал при подаче входного сигнала.

Концептуально среда моделирования состоит из двух частей – текстовой области и графского пространства. В ходе разработки были реализованы и отлажены алгоритм разбора введенной пользователем системы канонических уравнений, описывающих моделируемую систему, алгоритм отображения и перерисовки компонентов конечного автомата, представляющего исходную систему, алгоритм взаимодействия с элементами автомата, алгоритм симуляции работы моделируемого КА.

3 Реализация

Одной из задач данной работы является реализация моделируемого в предыдущей главе протокола OAuth2. В этой главе описываются детали реализации и выбранные для достижения этой задачи инструменты.

3.1 Keycloak

В качестве языка для разработки приложения, реализующего протокол OAuth2, был выбран язык Java. Как было сказано выше, одним из достоинств языка является широкий спектр подключаемых модулей, начиная от библиотеки, заканчивая организацией взаимодействия с другими приложениями, а также легкость и простота конфигурации подключаемых модулей.

В качестве основного средства, используемого для реализации OAuth2, был выбран Keycloak [16]. Keycloak — это самостоятельное, разворачивающееся приложение, представляющее собой сервер авторизации, который хранит информацию о пользователях, зарегистрированных в разрабатываемом приложении. Также Keycloak можно назвать идентификационным брокером, речь о которых шла в первой главе [17].

Keycloak самостоятельно разворачивается на каком-то хосту, прослушивающий какой-то порт. Взаимодействие с ним осуществляется по сети с помощью HTTPS по спецификации REST.

Keycloak является SSO (Single Sign On) решением для собственной реализации протоколов OAuth2 и OpenID Connect (OIDC) в разрабатываемых приложениях. Иными словами, с помощью Keycloak разрабатываемые вами приложения смогут поддерживать OAuth2 и OIDC.

Возможности Keycloak:

- single-sign-on и single-sign out для браузерных приложений,
- поддержка OpenID/OAuth 2.0/SAML,
- identity Brokering – аутентификация с помощью внешних OpenID Connect или SAML идентификационных провайдеров,

- social login – поддержка Google, GitHub, Facebook, Twitter для идентификации пользователей,
- user federation – синхронизация пользователей из LDAP и Active Directory серверов и других идентификационных провайдеров,
- kerberos bridge – использование Kerberos сервера для автоматической аутентификации пользователей,
- admin console — для единого управления настройками и параметрами решения через Web,
- account management console – для самостоятельного управления профилем пользователей,
- кастомизация решения на основе фирменного стиля компании,
- 2FA authentication – поддержка TOTP/HOTP с помощью Google Authenticator или FreeOTP,
- login Flows – возможна саморегистрация пользователей, восстановление и сброс пароля и прочие,
- session management – администраторы могут управлять из единой точки сессиями пользователей,
- token mappers – привязка атрибутов пользователей, ролей и иных требуемых атрибутов в токены,
- гибкое управление политиками через realm, application и пользователей,
- cors support – клиентские адаптеры имеют встроенную поддержку CORS,
- service provider interfaces (SPI) – большое количество SPI, позволяющих настраивать различные аспекты работы сервера: потоки аутентификации, идентификационных провайдеров, сопоставление протоколов и многое другое,
- клиентские адаптеры для JavaScript applications, WildFly, JBoss EAP, Fuse, Tomcat, Jetty, Spring,

- поддержка работы с различными приложениями, поддерживающими OpenID Connect Relying Party library или SAML 2.0 Service Provider Library,
- возможность расширения с использованием plugins.

3.2 Разработка приложения

Для достижения поставленных целей разработано приложение, реализующее протоколы OAuth2 и OIDC с помощью идентификационного брокера Keycloak.

3.2.1 Развертывание и настройка Keycloak

Для простоты развертывания, Keycloak разворачивается на локальной машине с помощью Docker-контейнера. Для этого достаточно установить Docker и передать через командную строку имя образа. Команда для запуска контейнера представлена на рисунке 36.

```
Microsoft Windows [Version 10.0.19044.1706]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

C:\Users\sbori>docker run -p 8484:8080 -e KEYCLOAK_USER=admin -e KEYCLOAK_PASSWORD=admin jboss/keycloak
```

Рисунок 36 - Запуск контейнера Keycloak

После успешного развертывания административная консоль Keycloak будет доступна по <http://localhost:8484/auth>. Результат представлен на рисунке 37.

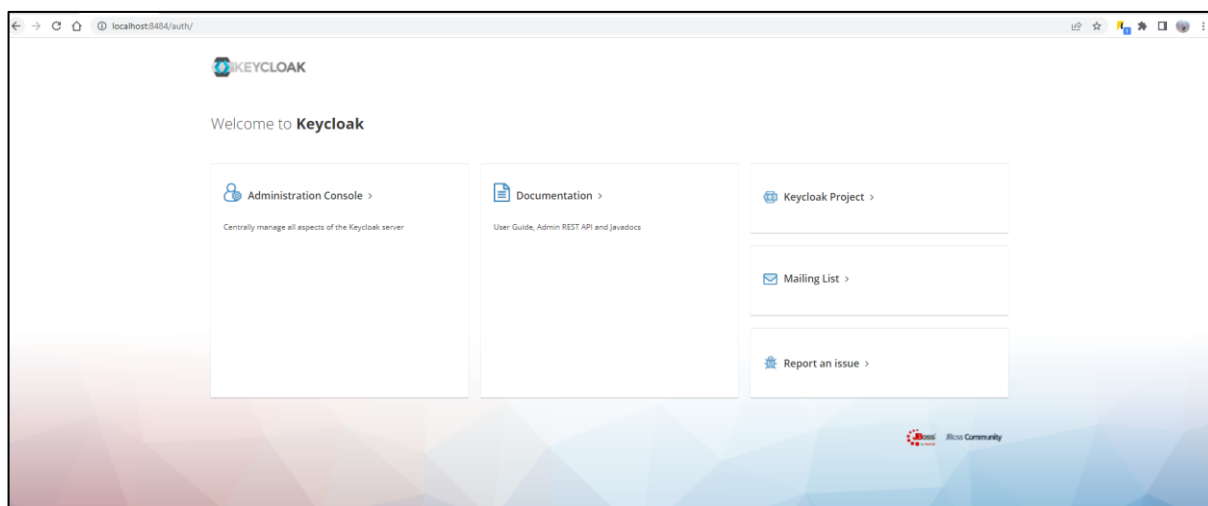


Рисунок 37 - Стартовая консоль

После перехода в консоль требуется создать realm - раздел в Keycloak, с которым будет работать приложение. Другими словами Realm – область

безопасности, которая определяется и настраивается в KeyCloak. Realm'ы в KeyCloak позволяют создавать несколько различных конфигураций безопасности.

Все настройки будут производиться в рамках созданного Realm: как будет проходить аутентификация, какие приложения будут ее использовать, какие пользователи могут проходить аутентификацию, какие атрибуты будут возвращаться и т.д. Результат создания Realm приведен на рисунке 38.

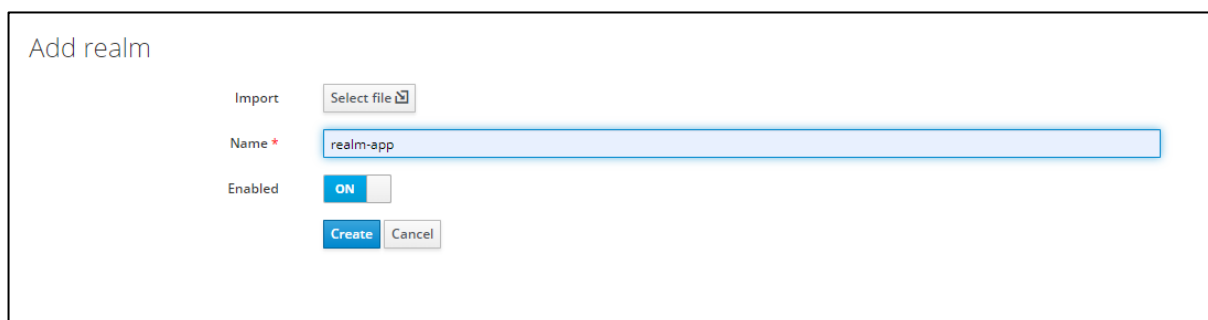


Рисунок 38 - Создание Realm

Результат настройки на рисунке 39.

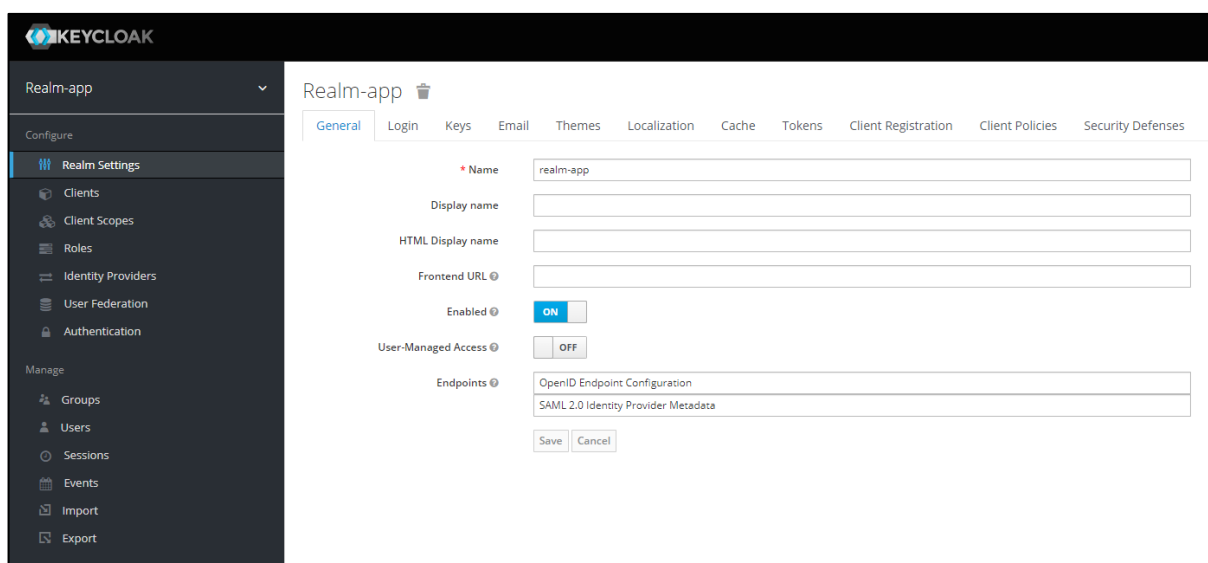


Рисунок 39 - Настройка Realm

Далее создадим клиента. Клиент представляет собой разрабатываемое приложение, которое будет запрашивать авторизацию у Keycloak. Результат представлен на рисунках 40, 41.

Clients > Add Client

Add Client

Import

Client ID *

Client Protocol

Root URL

Рисунок 40 - Создание клиента

Keycloak-app

Settings Credentials Keys Roles Client Scopes Mappers Scope Authorization Revocation Sessions Offline Acc

Client ID

Name

Description

Enabled ☒

Always Display in Console ☐

Consent Required ☐

Login Theme

Client Protocol

Access Type

Standard Flow Enabled ☒

Implicit Flow Enabled ☐

Direct Access Grants Enabled ☒

Service Accounts Enabled ☒

OAuth 2.0 Device Authorization Grant Enabled ☐

OIDC CIBA Grant Enabled ☐

Authorization Enabled ☒

Front Channel Logout ☐

Root URL

* Valid Redirect URIs

Base URL

Рисунок 41 - Настройка клиента

На странице настройки приложений есть множество опций. В нашем случае интересны следующие:

- client protocol - протокол авторизации – OIDC,

- standard flow enabled - метод авторизации,
- access type - имеет ли приложение client_secret,
- authorization enabled - активирована ли возможность авторизации,
- valid redirect url - Адрес перенаправления после успешной аутентификации.

Далее создадим 2 роли, используемые в приложении - user, admin. Результат представлен на рисунках 42 - 44.

Roles > Add Role

Add Role

* Role Name: user

Description: Роль для рядовых пользователей

Save Cancel

Рисунок 42 - Создание роли User

Roles > Add Role

Add Role

* Role Name: admin

Description: Роль для администраторов приложения

Save Cancel

Рисунок 43 - Создание роли Admin

Roles

Realm Roles Default Roles

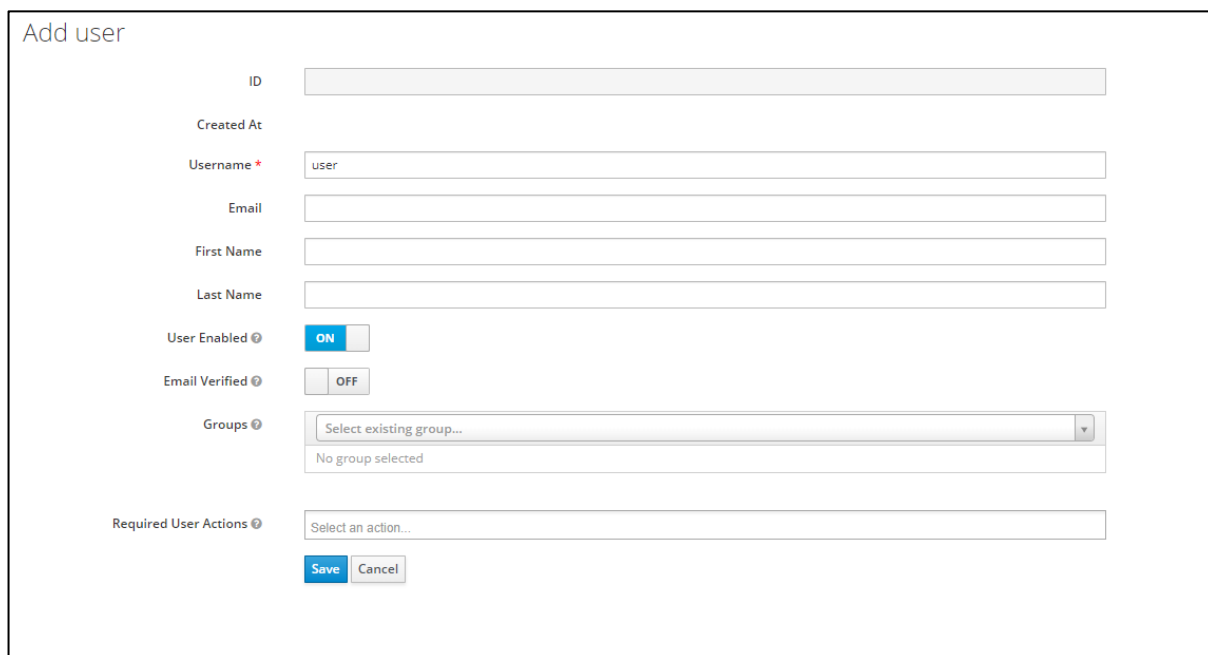
Search... View all roles Add Role

Role Name	Composite	Description	Actions
admin	False		Edit Delete
default-roles-real-app	True	\$(role_default-roles)	Edit Delete
offline_access	False	\$(role_offline-access)	Edit Delete
uma_authorization	False	\$(role_uma_authorization)	Edit Delete
user	False		Edit Delete

Рисунок 44 - Роли

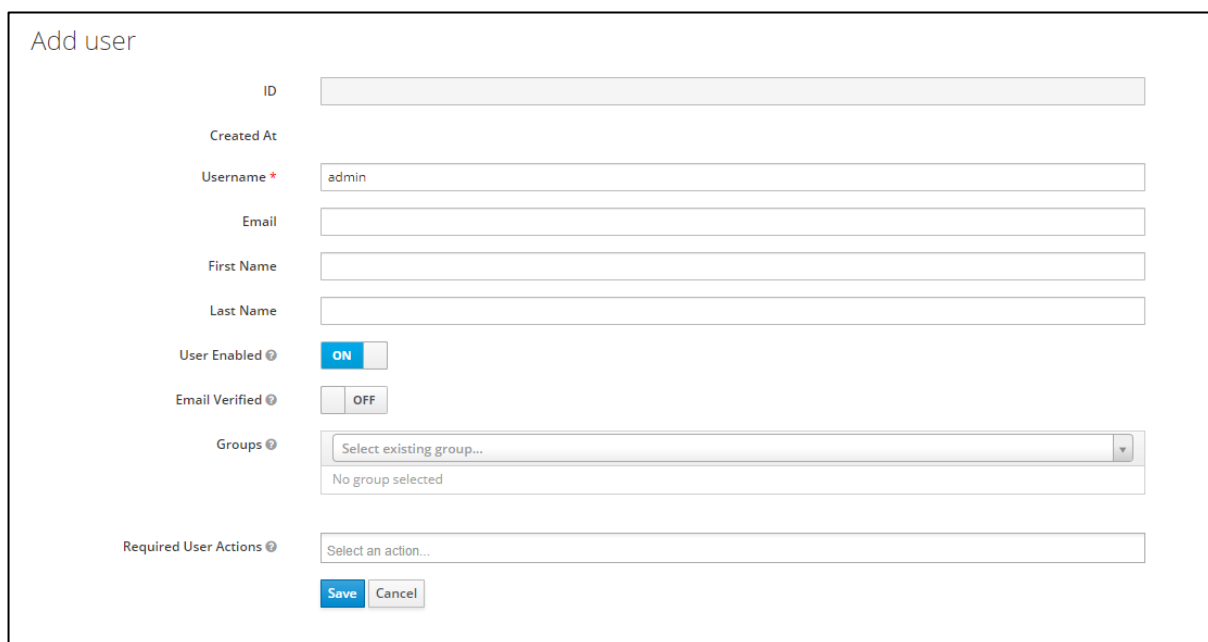
Также можно заметить, что помимо созданных в Keycloak предопределены другие роли, которые также можно использовать в клиентском приложении.

Далее создадим пользователей, которым присвоим созданные роли. Результат на рисунка 45, 46.



The screenshot shows a web form titled "Add user". It contains several input fields: "ID" (empty), "Created At" (empty), "Username *" (containing "user"), "Email" (empty), "First Name" (empty), and "Last Name" (empty). There are two toggle switches: "User Enabled" (set to "ON") and "Email Verified" (set to "OFF"). Below these is a "Groups" dropdown menu with the text "Select existing group..." and "No group selected". At the bottom is a "Required User Actions" dropdown menu with the text "Select an action...". There are "Save" and "Cancel" buttons at the bottom right.

Рисунок 45 - Создание пользователя User



The screenshot shows the same "Add user" form as in Figure 45, but with the "Username *" field containing "admin". All other fields and controls are identical: "ID" (empty), "Created At" (empty), "Email" (empty), "First Name" (empty), "Last Name" (empty), "User Enabled" (ON), "Email Verified" (OFF), "Groups" dropdown (No group selected), and "Required User Actions" dropdown (Select an action...). The "Save" and "Cancel" buttons are at the bottom right.

Рисунок 46 - Создание пользователя Admin

На вкладке Credentials зададим пароль для пользователя. Результат на рисунке 47.

The screenshot shows the 'Admin' interface with the 'Credentials' tab selected. Below the 'Manage Credentials' table, the 'Reset Password' form is visible. It includes fields for 'Password' and 'Password Confirmation', both containing the text 'pass'. There is a 'Temporary' toggle switch set to 'ON' and a 'Reset Password' button at the bottom.

Рисунок 47 - Установка пароля

На вкладке Role Mappings присвоим созданным пользователям созданные роли. Результат на рисунке 48.

The screenshot shows the 'Admin' interface with the 'Role Mappings' tab selected. It displays four panels: 'Realm Roles' (empty), 'Available Roles' (containing 'offline_access', 'uma_authorization', and 'user'), 'Assigned Roles' (containing 'admin' and 'default-roles-real-app'), and 'Effective Roles' (containing 'admin', 'default-roles-real-app', 'offline_access', and 'uma_authorization'). There are 'Add selected >' and '< Remove selected' buttons between the 'Available' and 'Assigned' panels. At the bottom, there is a 'Client Roles' section with a dropdown menu labeled 'Select a client...'.

Рисунок 48 - Привязка роли

На этом шаге минимальная настройка Keycloak закончена. Keycloak предоставляет большое множество настроек, поэтому процесс авторизации может быть настроен очень гибко.

3.2.2 Разработка клиентского приложения

Для разработки клиентского приложения был выбран фреймворк Spring Boot 2 ввиду своей популярности и простоты конфигурации.

Spring Boot — это проект, целью которого является упрощение создания приложений на основе Spring [18]. Он позволяет наиболее простым способом создать web-приложение, требуя от разработчиков минимум усилий по его настройке и написанию кода.

Разработчики Keycloak позаботились также о разработке адаптеров для подключения Keycloak к различным языкам и фреймворкам [19]. Spring Boot тоже в их числе.

Приложение было сконфигурировано следующим образом, подобно как на рисунке 49.

```
1  server:
2    port: 8081
3  spring:
4    mvc:
5      view:
6        prefix: /templates/
7        suffix: .html
8  keycloak:
9    auth-server-url: 'http://localhost:8484/auth'
10   realm: realm-app
11   resource: keycloak-app
12   credentials:
13     secret: hDfwKeZ39unR4qoiRzzj5p0tYtZlxraq
14   security-constraints:
15     - authRoles:
16       - user
17       securityCollections:
18         - patterns:
19           - /user
20     - authRoles:
21       - admin
22       securityCollections:
23         - patterns:
24           - /admin
```

Рисунок 49 - Конфигурация приложения

В поле `server.port` указывается порт на локальной машине, который будет прослушивать сервер.

Prefix и Suffix - расположение и расширение view-файлов, представляющих собой html-страницы, возвращающиеся в ответе пользователю.

Auth-server-url - адрес, на котором развернута консоль управления Keycloak.

Realm - пространство, область с определенными правилами аутентификации и авторизации, которые проходят все клиенты (приложения), зарегистрированные в этом realm.

Resource - имя клиента (приложения) на сервере Keycloak.

Credentials.secret - секрет приложения, ClientSecret по спецификации OAuth2

Security-constraints - правила, по которым осуществляется авторизация на сервере Keycloak, при обращении по определенным URL разрабатываемого приложения [20]. В данном случае ресурсы по URL /user могут быть доступны только пользователям с ролью user. Аналогично, ресурсы по URL /admin могут быть доступны только пользователям с ролью admin.

Процесс работы приложения продемонстрирован далее.

Шаг 1. В адресной строке браузера вводим адрес и порт, по которым доступно наше приложение. Результат на рисунке 50.

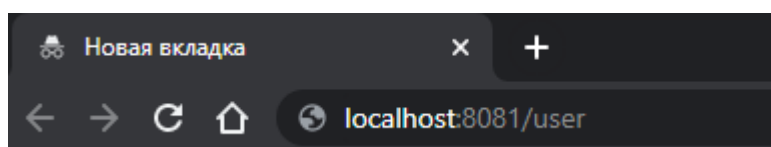


Рисунок 50 - Url приложения

Шаг 2. При переходе по введенному адресу происходит переход на сервер Keycloak. Результат на рисунке 51.

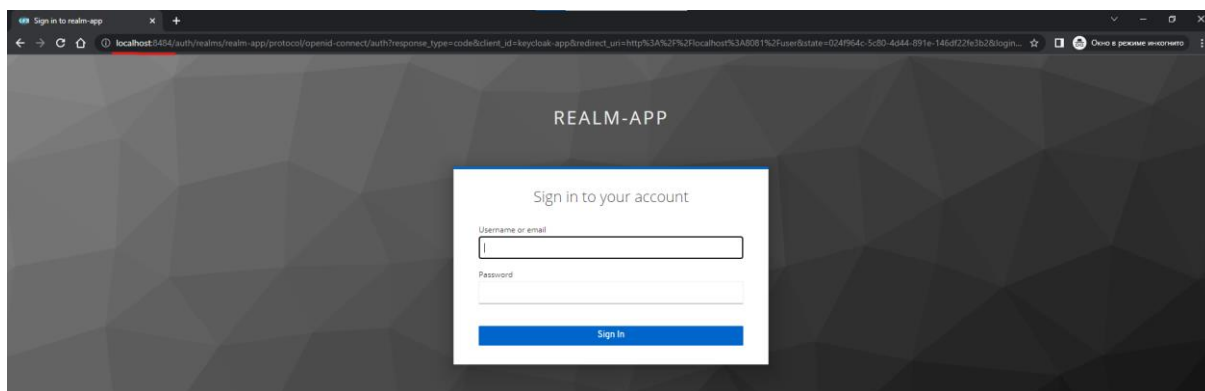


Рисунок 51 - Страница авторизации

Шаг 3. После авторизации пользователя происходит проверка на наличие у этого пользователя необходимой роли для доступа к запрашиваемому ресурсу. Если пользователь обладает такой ролью, то переход на соответствующую страницу. Результат на рисунке 52.

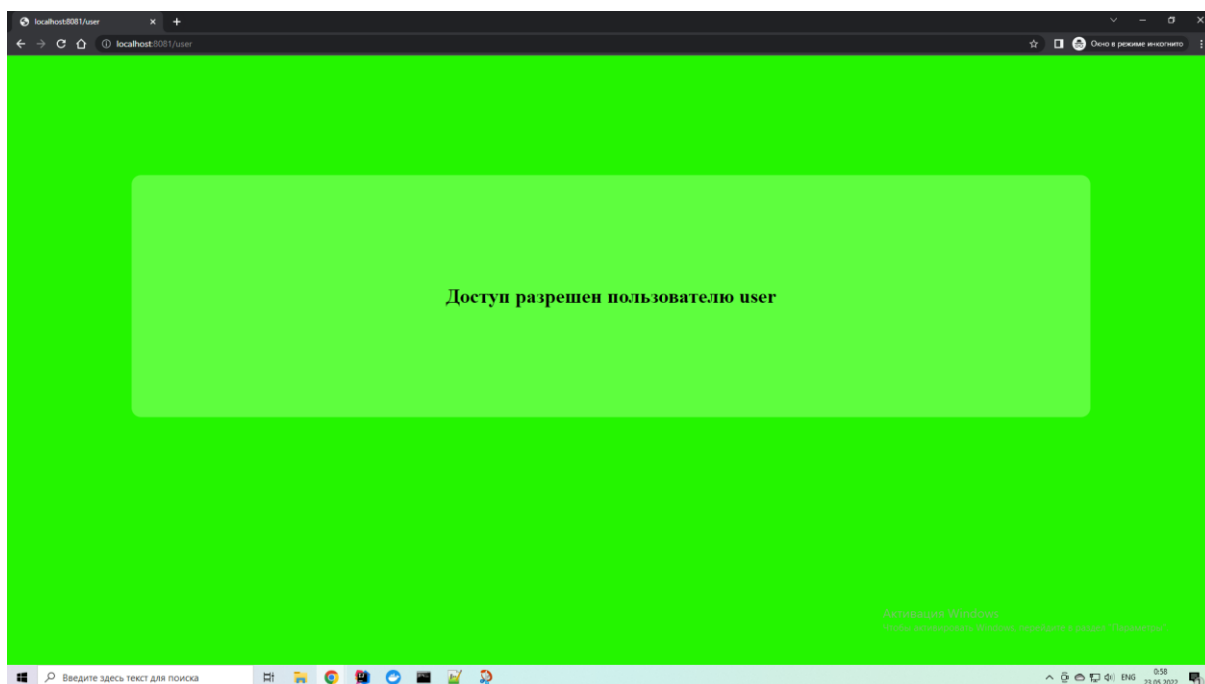


Рисунок 52 - Доступ разрешен

Шаг 4. Если у пользователя нет нужной роли, то происходит переход на страницу ошибки. Результат на рисунке 53.

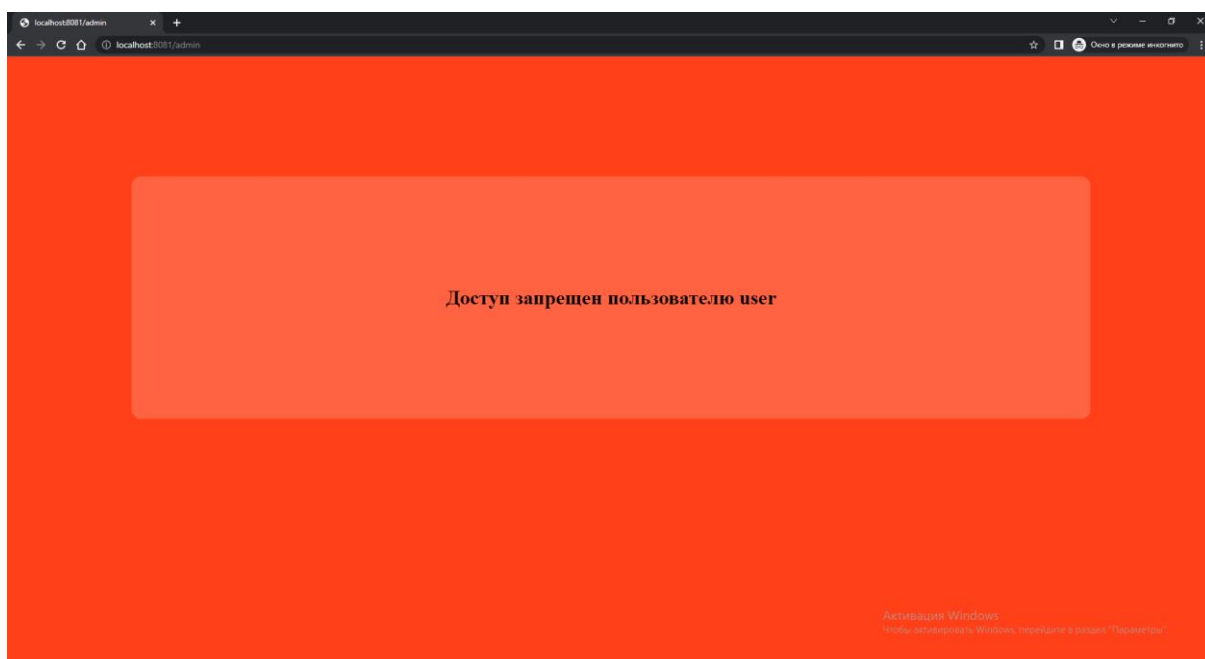


Рисунок 53 - Доступ запрещен

Добавим еще один контроллер, обрабатывающий REST-запросы по URL /api. По данному запросу предоставляется уже не html-страница, и информация в текстовом виде, в соответствии со спецификацией REST.

Представим, что по этому адресу будут обращаться другие приложения и сервисы.

Также защитим доступ и предоставим его только для пользователей с ролью User. Чтобы получить доступ для этого ресурса, необходимо запросить токен авторизации у сервера Keycloak по протоколу OAuth2.

Шаг 1. Для получения временного кода доступа (access_token) необходимо обратиться по запросу `http://localhost:8484/auth/realms/realm-app/protocol/openid-connect/auth?client_id=keycloak-app&response_type=code`.

Результат на рисунке 54.

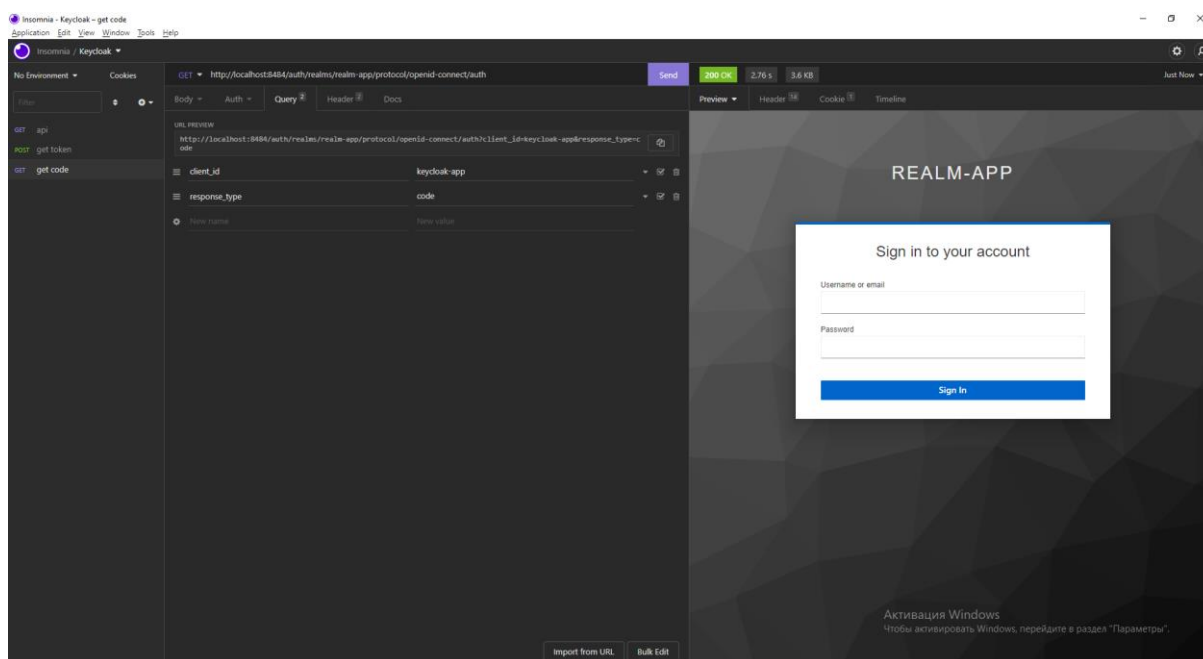


Рисунок 54 - Rest-запрос кода авторизации

Шаг 2. Если у предоставленного client_id еще нет активной сессии, то необходимо выполнить вход.

Шаг 3. После успешного входа, произойдет redirect на адрес, указанный в настройках keycloak-app (`http://localhost:8081/`), при этом к параметрам запроса добавится сгенерированный временный код. Результат на рисунке 55.

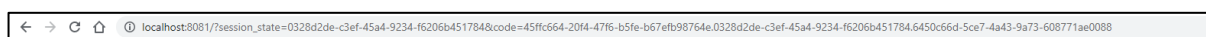


Рисунок 55 - Redirect-Url

Шаг 4. Для обмена этого кода на токен доступа нужно отправить post-запрос `http://localhost:8484/auth/realms/realm-app/protocol/openid-connect/token`,

предоставив при этом необходимые данные (client_id, client_secret, code).
Результат на рисунке 56.

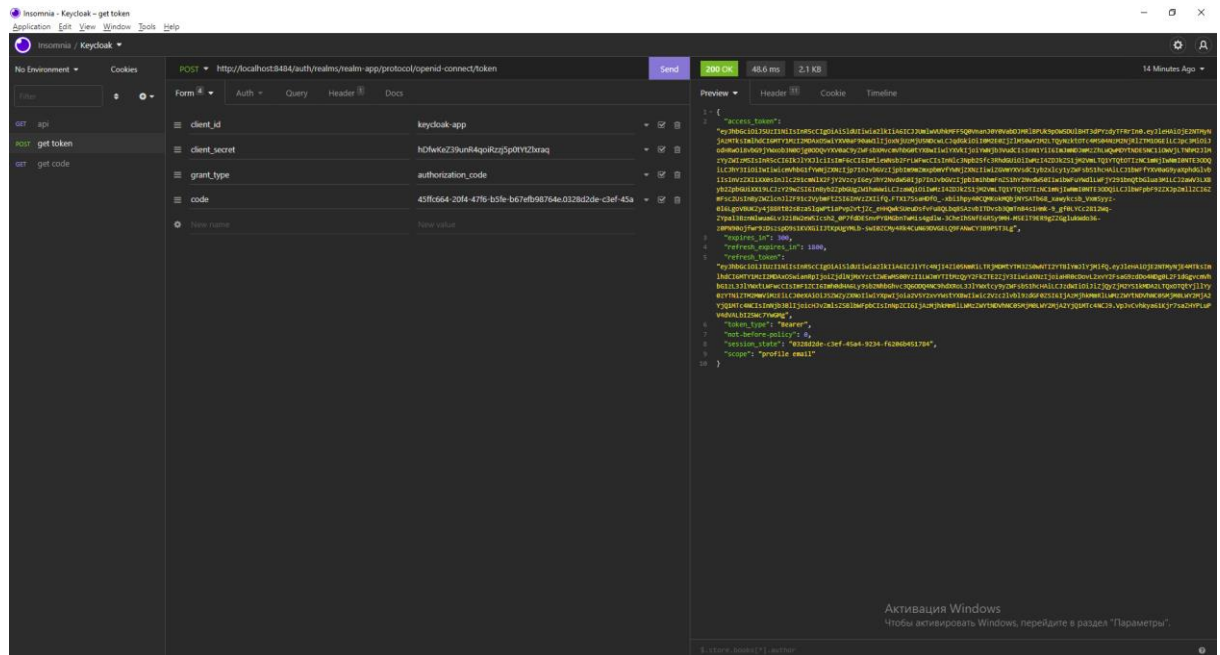


Рисунок 56 - Ответ, содержащий токен

Шаг 5. В ответе вернется токен доступа. Для доступа к защищенным ресурсам, токен необходимо передавать в заголовке Authorization каждого запроса. Результат на рисунке 57.

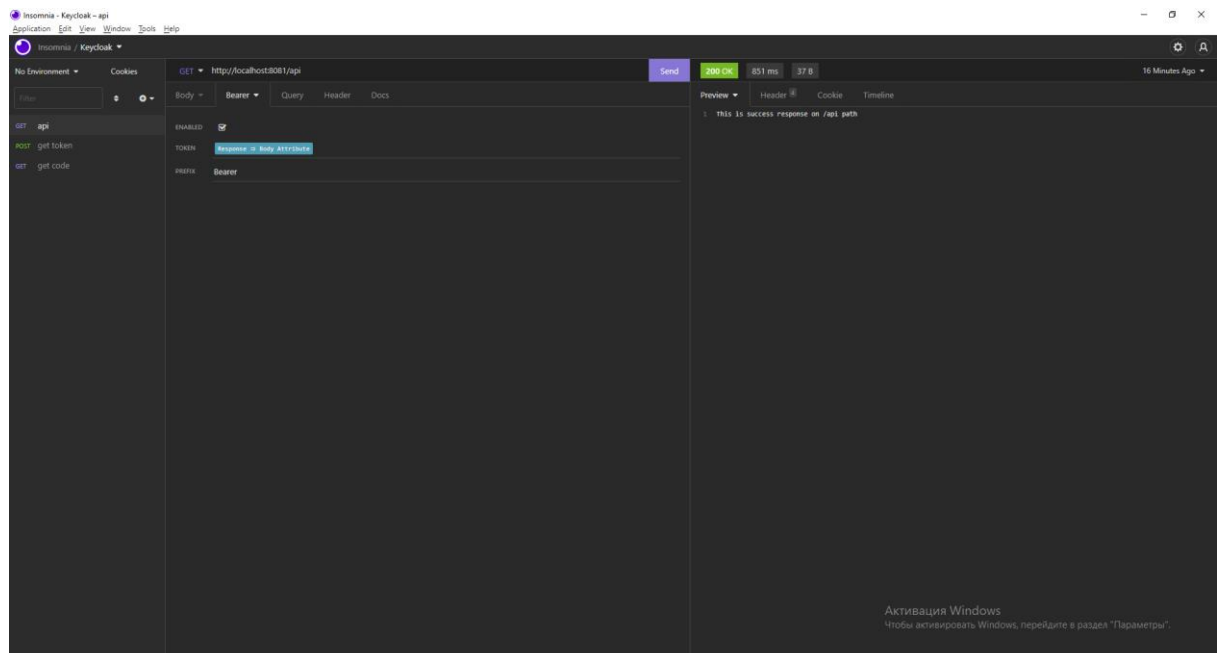


Рисунок 57 - Запрос ресурса с токеном

3.3 Выводы

В результате реализации исследуемого алгоритм безопасности было разработано приложение, реализующее этот алгоритм.

В качестве языка программирования был выбран язык Java из-за удобства использования и большого количества существующих для языка библиотек и модулей, а также фреймворк для разработки веб-серверов Spring Boot 2.

В качестве идентификационного брокера (сервера авторизации) был выбран Keycloak - сервис, реализующий протоколы OAuth2 и OIDC в разрабатываемых приложениях и осуществляющий аутентификацию и авторизацию пользователей.

ЗАКЛЮЧЕНИЕ

В ходе выполнения данной работы были приобретены навыки описания и моделирования распределенных информационных систем, навыки разработки оконных приложений с помощью фреймворка Windows Form и серверных приложений на Java с помощью фреймворка Spring Boot. Результатом выполнения работы стала среда моделирования поведения и симуляции работы распределенных информационных систем, а также приложение, реализующее протоколы OAuth2, OIDC.

В разработанной среде моделирования описание систем происходит с помощью конечного автомата, который задается через систему канонических уравнений.

Разработанная среда моделирования способна решать такие задачи, как проектирование сложных информационных систем, моделирование их поведения, общий анализ вычислительных систем, выявление узких и проблемных мест в их работе еще на стадии проектирования.

Основными достоинствами разработанной среды являются наличие удобного и понятного для использования интерфейса, ресурсоемкость и производительность.

Приложение «Инструментальная система темпорального анализа моделей параллельных вычислительных систем» было представлено на семинаре по электронным и сетевым технологиям MWENT-2020, проходившего в стенах Пензенского Государственного Университета. Результаты трудов семинара были проиндексированы рефератной базой данных SCOPUS.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Жук А.П. Защита информации / А.П. Жук, Е.П. Жук, О.М. Лепешкин. - Москва: ИЦ РИОР, 2021. - 400 с. - ISBN 978-5-369-01759-3.
2. Гришина Н. В. Организация комплексной системы защиты информации //М.: Гелиос АРВ. – 2007. – Т. 256.
3. Гришина Н. В. Организация комплексной системы защиты информации // М.: Гелиос АРВ, 2007. — 256 с, ил. ISBN 978-5-85438-171-0.
4. Sucasas V. et al. An OAuth2-based protocol with strong user privacy preservation for smart city mobile e-Health apps //2016 IEEE International Conference on Communications (ICC). – IEEE, 2016. – С. 1-6.
5. Siriwardena P. OpenID Connect (OIDC) //Advanced API Security. – Apress, Berkeley, CA, 2020. – С. 129-155.
6. Yumerov B., Ivanova G. OIDC Authentication for Educational Purposes and Solving Problems for Localization of Faults in Combinational Circuits //Proceedings of the Computational Methods in Systems and Software. – Springer, Cham, 2020. – С. 419-429.
7. Грекул В. И., Денищенко Г. Н., Коровкина Н. Л. Проектирование информационных систем //М.: РГГУ. – 2018. – Т. 224.
8. Хмелёв Д. В. Распознавание автора текста с использованием цепей АА Маркова //Вестник МГУ, сер. – 1999. – Т. 9. – С. 115-126.
9. Мальков М. В., Малыгина С. Н. Сети Петри и моделирование //Труды Кольского научного центра РАН. – 2010. – №. 3. – С. 35-40.
10. Глушков В. М. Абстрактная теория автоматов «Успехи математических наук». – 1961. – Т. 16. – №. 5 (101. – С. 3-62.
11. Ожиганов А. А. Теория автоматов «Учебное пособие». –СПб: НИУ ИТМО. – 2013.
12. Troelsen A. Programming with Windows Form Controls //Visual Basic. NET and the. NET Platform: An Advanced Guide. – Apress, Berkeley, CA, 2002. – С. 589-675.

13. Гамма, Хелм Р., Джонсон Р., Влиссидес Дж. Приемы объектно-ориентированного проектирования Паттерны проектирования. — СПб.: «Питер», 2001. — С. 18-20.
14. Лафоре Р. Объектно-ориентированное программирование в C++. — «Издательский дом Питер», 2013.
15. Романчук В. А., Ручкин В. Н. Оценка результатов моделирования вычислительных систем на базе нейропроцессоров «Известия тульского государственного университета. Технические науки». — 2013. — №. 9-2.
16. Divyabharathi D. N., Cholli N. G. A review on identity and access management server (keycloak) //International Journal of Security and Privacy in Pervasive Computing (IJSPPC). — 2020. — Т. 12. — №. 3. — С. 46-53.
17. Chatterjee A., Prinz A. Applying spring security framework with keycloak-based OAuth2 to protect microservice architecture APIs: A case study //Sensors. — 2022. — Т. 22. — №. 5. — С. 1703.
18. Webb P. et al. Spring boot reference guide //Part IV. Spring Boot features. — 2013. — Т. 24.
19. Nguyen Q., Baker O. F. Applying Spring Security Framework and OAuth2 To Protect Microservice Architecture API //J. Softw. — 2019. — Т. 14. — №. 6. — С. 257-264.
20. Alex B. et al. Spring Security Reference //URL [https://docs. spring. io/springsecurity/site/docs/current/reference/htmlsingle/](https://docs.spring.io/springsecurity/site/docs/current/reference/htmlsingle/). [utoljára megtekintve: 2017. 04. 21.]. — 2004. — Т. 12.

ПРИЛОЖЕНИЕ А

Листинги среды моделирования

Файл «Parser.cs»

```
using System;
using System.Text;
using System.IO;
using sku_to_smv.Properties;
using System.Text.RegularExpressions;
using sku_to_smv.src;
using System.Collections.Generic;

namespace sku_to_smv
{
    public enum parceResult { PARSE_OK, PARCE_ERROR };

    public class Parser
    {
        enum state { H, S1, S2, S3, S4, S5, S6, S7, S8, S9, S10, S11, S12, S13, ERR, END,
EXT, OPT, OPT_END, TIME, O1, O2, O3, O4, O5 };
        enum parseStates { TARGET_STATE, SOURCE_STATE, TIME, SIGNAL,
OPERATION, INPUT_SIGNALS, SKU }

        int RCount;           //Количество правил полученных после анализа
        bool Inv;
        bool isSKU;
        bool isOUT;

        public List<Rule> Rules;           //Массив правил, строящийся при анализе
        public Rule[] OutputRules;        //Массив правил, строящийся при анализе
        public string[] LocalStates;       //Список локальных состояний
        public string[] Inputs;            //Список входных сигналов
        public string[] RecovStates;       //Список повторновходимых состояний
        public string[] Outputs;          //Список выходных сигналов
        public bool signalsParsed;
        public List<Signal> signals;
        public List<State> states;
        public Parser()
        {
            //Инициализируем массивы
            Rules = new List<Rule>();
            LocalStates = new string[0];
            Inputs = new string[0];
            Outputs = new string[0];
            signals = new List<Signal>();
            states = new List<State>();
            RecovStates = new string[0];
            RCount = 0;
            signalsParsed = false;
        }
    }
    /// <summary>
```

```

/// Добавляет новое правило в массив
/// </summary>
/*private void AddRule()
{
    Array.Resize(ref Rules, Rules.Length + 1);
    Rules[Rules.Length - 1] = new Rule();
    RCount++;
}
private void AddOutRule()
{
    Array.Resize(ref OutputRules, OutputRules.Length + 1);
    OutputRules[Rules.Length - 1] = new Rule();
    //RCount++;
}*/

private bool isExistsState(String stateName)
{
    foreach (State state in this.states)
    {
        if (state.Name.Equals(stateName)) return true;
    }
    return false;
}

private State getStateByName(String name)
{
    foreach (State state in this.states)
    {
        if (state.Name.Equals(name)) return state;
    }
    throw new Exception("State with name " + name + " not found");
}

private Signal getSignalByName(String name)
{
    foreach (Signal signal in this.signals)
    {
        if (signal.name.Equals(name)) return signal;
    }
    throw new Exception("State with name " + name + " not found");
}
/// <summary>
/// Запуск парсера
/// </summary>
/// <param name="InputString">Текст для разбора</param>
/// <returns>Результат разбора типа parseResult</returns>
public parseResult ParseStart(String InputString)
{
    int k = 0;
    string STR;
    string inputSTR = InputString;
    int[] EndOfRules = new int[0];

```

```

bool b_Comment = false;
isSKU = true;
isOUT = false;
StringBuilder sb1 = new StringBuilder();
//Rules = new Rule[0];
LocalStates = new string[0];
Inputs = new string[0];
Outputs = new string[0];
RecovStates = new string[0];
GC.Collect();

//Убираем пробелы, знаки табуляции, перехода на новую строку и
комментарии
for (int i = 0; i < inputSTR.Length; i++)
{
    if (inputSTR[i] == '#' && !b_Comment) b_Comment = true;
    else if (inputSTR[i] == '#' && b_Comment) b_Comment = false;
    if (inputSTR[i] != 32 && inputSTR[i] != '\t' && inputSTR[i] != '\n' &&
!b_Comment && inputSTR[i] != '#')
    {
        sb1.Append(inputSTR[i]);
    }
}
inputSTR = sb1.ToString();
//Поиск всех знаков ';'
for (int i = 0; i < inputSTR.Length; i++)
{
    if (inputSTR[i] == ';')
    {
        Array.Resize(ref EndOfRules, EndOfRules.Length + 1);
        EndOfRules[EndOfRules.Length - 1] = i;
    }
}
//Переводим в нижний регистр
inputSTR = inputSTR.ToLower();

sb1.Clear();
k = 0;
for (int i = 0; i < EndOfRules.Length; i++)
{
    //Выделение одного правила
    STR = inputSTR.Substring(k, EndOfRules[i] - k + 1);
    k = EndOfRules[i] + 1;
    //Разбор правила
    if (parce(STR) == parceResult.PARCE_ERROR)
    {
        return parceResult.PARCE_ERROR;
    }
}
//Составление списка локальных состояний//
//LocalStates = new string[Rules.Length];
/*for (int i = 0; i < Rules.Length; i++)

```

```

    {
        if (Rules[i].Elems[0].Output)
        {
            Array.Resize(ref Outputs, Outputs.Length + 1);
            Outputs[Outputs.Length - 1] = Rules[i].Elems[0].Value;
        }
        else
        {
            Array.Resize(ref LocalStates, LocalStates.Length + 1);
            LocalStates[LocalStates.Length - 1] = Rules[i].Elems[0].Value;
        }
    }
    SearchForInputs();//Составление списка входных сигналов//
    SearchForRecov();//Составление списка повторновходимых состояний//*/

    return parceResult.PARSE_OK;
}
/// <summary>
/// Разбор очередной строки
/// </summary>
/// <param name="InputString">Строка для разбора</param>
/// <param name="StartState">Начальное состояние парсера</param>
/// <returns>Результат разбора типа parceResult</returns>
/*private parceResult Parse(String InputString, state StartState)
{
    state st1 = StartState;
    int k = 0;
    string tmp;
    string time = "";
    for (int i = 0; i < InputString.Length; i++)
    {
        if (!signalsParsed) parce(InputString);
        switch (st1)
        {
            //Буква
            case state.H:
                if (InputString[i] == '[')
                    st1 = state.OPT;
                else if (Regex.IsMatch(InputString[i].ToString(), "[a-z]"))
                {
                    k = i;
                    //Rules[Rules.Length - 1].output = isOUT;
                    st1 = (isSKU) ? state.S1 : state.O1;
                }
                else st1 = state.ERR;
                break;
            //Если новая секция
            case state.OPT:
                if (InputString[i] == 's')
                {
                    isSKU = true;
                    isOUT = false;

```

```

        //Rules[Rules.Length - 1].output = false;
        st1 = state.OPT_END;
    }
    else if (InputString[i] == 'o')
    {
        isSKU = false;
        isOUT = true;
        //Rules[Rules.Length - 1].output = true;
        st1 = state.OPT_END;
    }
    else st1 = state.ERR;
    break;
//Ожидаем конца определения секции
case state.OPT_END:
    if (InputString[i] == ']')
        st1 = state.H;
    break;
case state.O1:
    if (Regex.IsMatch(InputString[i].ToString(), "[a-z0-9]"))
    {
        st1 = state.O2;
    }
    else if (InputString[i] == '{')
    {
        st1 = state.TIME;
    }
    else st1 = state.ERR;
    break;
case state.O2:
    if (InputString[i] == '<')
    {
        st1 = state.O3;
        tmp = InputString.Substring(k, i - k);
        if (tmp.Contains("{")) tmp = tmp.Substring(0, tmp.IndexOf("{"));
        //Rules[Rules.Length - 1].AddData("State", tmp, false, isOUT);
        k = i + 1;
    }
    else if (Regex.IsMatch(InputString[i].ToString(), "[a-z0-9]"))
    {
        st1 = state.O2;
    }
    else st1 = state.ERR;
    break;
case state.O3:
    if (InputString[i] == '=')
    {
        st1 = state.O4;
        //Rules[Rules.Length - 1].AddData("<=", "<=", false, isOUT);
        k = i + 1;
    }
    else st1 = state.ERR;
    break;

```

```

case state.O4:
    if (Regex.IsMatch(InputString[i].ToString(), "[a-z]"))
    {
        k = i;
        st1 = state.O5;
    }
    else st1 = state.ERR;
    break;
case state.O5:
    if (Regex.IsMatch(InputString[i].ToString(), "[a-z0-9]"))
    {
        st1 = state.O5;
    }
    else if (InputString[i] == ';')
    {
        tmp = InputString.Substring(k, i - k);
        if (tmp.Contains("{")) tmp = tmp.Substring(0, tmp.IndexOf("{"));
        Rules[Rules.Length - 1].AddData("State", tmp, false, isOUT);
        st1 = state.END;
    }
    else st1 = state.ERR;
    break;
//Буква или цифра или '(' или '='
case state.S1:
    if (Regex.IsMatch(InputString[i].ToString(), "[a-z0-9]"))
    {
        st1 = state.S1;
    }
    else if (InputString[i] == '(')
    {
        st1 = state.S2;
        tmp = InputString.Substring(k, i - k);
        if (tmp.Contains("{")) tmp = tmp.Substring(0, tmp.IndexOf("{"));
        Rules[Rules.Length - 1].AddData("State", tmp, false, isOUT);
        k = i + 1;
    }
    else if (InputString[i] == '=')
    {
        st1 = state.S7;
        tmp = InputString.Substring(k, i - k);
        if (tmp.Contains("{")) tmp = tmp.Substring(0, tmp.IndexOf("{"));
        Rules[Rules.Length - 1].AddData("State", tmp, false, isOUT);
        Rules[Rules.Length - 1].AddData("=", "=", false);
        k = i + 1;
    }
    else st1 = state.ERR;
    break;
case state.S2:
    if (InputString[i] == 't')
    {
        st1 = state.S3;
    }

```

```

        else st1 = state.ERR;
        break;
    case state.S3:
        if (InputString[i] == '+')
        {
            st1 = state.S4;
        }
        else st1 = state.ERR;
        break;
    case state.S4:
        if (InputString[i] == '1')
        {
            st1 = state.S5;
        }
        else st1 = state.ERR;
        break;
    case state.S5:
        if (InputString[i] == ')')
        {
            Rules[Rules.Length - 1].AddData("t+1", "(t+1)", false);
            k = i + 1;
            st1 = state.S6;
        }
        else st1 = state.ERR;
        break;
    case state.S6:
        if (InputString[i] == '=')
        {
            Rules[Rules.Length - 1].AddData("=", "=", false);
            k = i + 1;
            st1 = state.S7;
        }
        else st1 = state.ERR;
        break;
    //Буква или '(' или '~'
    case state.S7:
        if (Regex.IsMatch(InputString[i].ToString(), "[0-9]"))
        {
            st1 = state.S7;
        }
        else if (Regex.IsMatch(InputString[i].ToString(), "[a-z]"))
        {
            k = i;
            Inv = false;
            st1 = state.S11;
        }
        else if (InputString[i] == '(')
        {
            Inv = false;
            Rules[Rules.Length - 1].AddData("(", "(", Inv);
            int n = 0;
            for (int j = i; j < InputString.Length; j++)

```

```

        {
            if (InputString[j] == '(')
            {
                n++;
            }
            if (InputString[j] == ')')
            {
                n--;
            }
            if (n == 0)
            {
                k = j;
                break;
            }
        }
        if (Parse(InputString.Substring(i + 1, k - i - 1) + ";", state.S7) ==
parceResult.PARCE_ERROR)
        {
            st1 = state.ERR;
        }
        i = k - 1;
        st1 = state.S8;
    }
    else if (InputString[i] == '~')
    {
        st1 = state.S10;
        Inv = true;
    }
    else st1 = state.ERR;
    break;
// ')'
case state.S8:
    if (InputString[i] == ')')
    {
        Rules[Rules.Length - 1].AddData("", ""), false);
        st1 = state.S9;
    }
    else st1 = state.ERR;
    break;
// '&' или '|' или ';'
case state.S9:
    if (InputString[i] == '&')
    {
        Rules[Rules.Length - 1].AddData("&", "&", false);
        st1 = state.S7;
    }
    else if (InputString[i] == '|')
    {
        Rules[Rules.Length - 1].AddData("|", "|", false);
        st1 = state.S7;
    }
    else if (InputString[i] == ';')

```

```

    {
        st1 = state.END;
    }
    break;
// Буква или '(' или
case state.S10:
    if (Regex.IsMatch(InputString[i].ToString(), "[a-z]"))
    {
        k = i;
        st1 = state.S11;
    }
    else if (InputString[i] == '(')
    {
        Inv = false;
        Rules[Rules.Length - 1].AddData("(", "(", Inv);
        int n = 0;
        for (int j = i; j < InputString.Length; j++)
        {
            if (InputString[j] == '(')
            {
                n++;
            }
            if (InputString[j] == ')')
            {
                n--;
            }
            if (n == 0)
            {
                k = j;
                break;
            }
        }
        if (Parse(InputString.Substring(i + 1, k - i - 1) + ";", state.S7) ==
parceResult.PARCE_ERROR)
        {
            st1 = state.ERR;
        }
        i = k - 1;
        st1 = state.S8;
    }
    break;
case state.S11:
    if (Regex.IsMatch(InputString[i].ToString(), "[a-z0-9]"))
    {
        st1 = state.S11;
    }
    else if (InputString[i] == '{')
    {
        st1 = state.TIME;
    }
    else if (InputString[i] == '&')
    {

```

```

        st1 = state.S7;
        string stateName = InputString.Substring(k, i - k);
        if (stateName.Contains("{")) stateName = stateName.Substring(0,
stateName.IndexOf("{"));
        Rules[Rules.Length - 1].AddData("State", stateName, Inv);
        Rules[Rules.Length - 1].AddData("&", "&", false);
    }
    else if (InputString[i] == '|')
    {
        st1 = state.S7;
        string stateName = InputString.Substring(k, i - k);
        if (stateName.Contains("{")) stateName = stateName.Substring(0,
stateName.IndexOf("{"));
        Rules[Rules.Length - 1].AddData("State", stateName, Inv);
        Rules[Rules.Length - 1].AddData("|", "|", false);
    }
    else if (InputString[i] == ';')
    {
        st1 = state.END;
        string stateName = InputString.Substring(k, i - k);
        if (stateName.Contains("{")) stateName = stateName.Substring(0,
stateName.IndexOf("{"));
        Rules[Rules.Length - 1].AddData("State", stateName, Inv);
    }
    else st1 = state.ERR;
    break;
case state.TIME:
    if (InputString[i] == '}')
    {
        Rules[Rules.Length - 1].AddData("TimeTransfer", time, Inv);
        st1 = state.S11;
        time = "";
        break;
    }
    time += InputString[i].ToString();
    st1 = state.TIME;
    break;
case state.ERR: //PrintText("Ошибка разбора");
    //st1 = state.EXT;
    break;
case state.END: //j++;
    //AddRule();
    //st1 = state.H;
    break;
case state.EXT:
    break;
    }
}
if (st1 == state.ERR)
{
    return parceResult.PARCE_ERROR;
}

```

79

```

    }
    break;
}
case 2:
{
    MatchCollection matches = stateReg.Matches(match.Groups[i].Value);
    foreach (Match m in matches)
    {
        for (int j = 1; j < m.Groups.Count; j++)
        {
            String stateName = m.Groups[j].Value;
            State beginState;
            if (isExistsState(stateName))
            {
                beginState = getStateByName(stateName);
            }
            else
            {
                beginState = new State();
                beginState.Name = stateName;
                states.Add(beginState);
            }
            beginStates.Add(beginState);
        }
    }
    break;
}
case 3:
{
    Match timeMatch = timeReg.Match(match.Groups[i].Value);
    for (int j = 1; j < timeMatch.Groups.Count; j++)
    {
        timeMark = new Time(timeMatch.Groups[j].Value);
    }
    MatchCollection matches = signalReg.Matches(match.Groups[i].Value);
    foreach (Match m in matches)
    {
        for (int j = 1; j < m.Groups.Count; j++)
        {
            String signalName = m.Groups[j].Value;
            try
            {
                if (signalName.Contains("!"))
                {
                    signalName = signalName.Substring(1, signalName.Length -
1);
                    inventeredSignals.Add(getSignalByName(signalName));
                }
                signals.Add(getSignalByName(signalName));
            }
            catch (Exception)
            {

```

```

        return parceResult.PARCE_ERROR;
    }
    }
    }
    break;
}
}
}
if (endState == null)
{
    return parceResult.PARCE_ERROR;
}
if (signals.Count != 0)
{
    foreach (Signal signal in signals)
    {
        foreach (State state in beginStates)
        {
            Rule rule = new Rule(state, endState, signal, timeMark,
inventeredsignals.Contains(signal));
            rules.Add(rule);
        }
    }
}
else
{
    foreach (State state in beginStates)
    {
        Rule rule = new Rule(state, endState, null, timeMark, false);
        rules.Add(rule);
    }
}

this.Rules.AddRange(rules);
return parceResult.PARSE_OK;
}

/// <summary>
/// Функция поиска входных сигналов
/// </summary>
/*private void SearchForInputs()
{
    bool NotInput = false;
    bool Exists = false;
    for (int i = 0; i < Rules.Length; i++)
    {
        for (int j = 0; j < Rules[i].Elems.Length; j++)
        {
            if (Rules[i].Elems[j].Type == "State")
            {
                for (int k = 0; k < LocalStates.Length; k++)
                {

```

```

        if (Rules[i].Elems[j].Value == LocalStates[k])
        {
            NotInput = true;
            break;
        }
    }
    for (int k = 0; k < Inputs.Length; k++)
    {
        if (Rules[i].Elems[j].Value == Inputs[k])
        {
            Exists = true;
            Rules[i].Elems[j].Local = false;
            break;
        }
    }
    if (!NotInput && !Exists && !Rules[i].Elems[j].Output)
    {
        Array.Resize(ref Inputs, Inputs.Length + 1);
        Inputs[Inputs.Length - 1] = Rules[i].Elems[j].Value;
        Rules[i].Elems[j].Local = false;
    }
    NotInput = false;
    Exists = false;
}
}
}
}*/
/// <summary>
/// Функция поиска повторновходимых состояний
/// </summary>
/*private void SearchForRecov()
{
    for (int i = 0; i < Rules.Length; i++)
    {
        for (int j = 1; j < Rules[i].Elems.Length; j++)
        {
            if (Rules[i].Elems[0].Value == Rules[i].Elems[j].Value)
            {
                Array.Resize(ref RecovStates, RecovStates.Length + 1);
                RecovStates[RecovStates.Length - 1] = Rules[i].Elems[j].Value;
                break;
            }
        }
    }
}
}*/
/// <summary>
/// Функция сохранения результата в SMV
/// </summary>
/// <param name="Path">Путь для сохранения</param>
public void SaveToSMV(string Path)
{
    StreamWriter sw = new StreamWriter(Path, false);

```

```

sw.Write("module main(");
sw.Flush();
for (int i = 0; i < Inputs.Length; i++)
{
    sw.Write(Inputs[i]);
    if ((i != Inputs.Length - 1) || Outputs.Length > 0)
    {
        sw.Write(",");
    }
    sw.Flush();
}
for (int i = 0; i < Outputs.Length; i++)
{
    sw.Write(Outputs[i]);
    if (i != Outputs.Length - 1)
    {
        sw.Write(",");
    }
    sw.Flush();
}
sw.Write("\r\n{\r\n");
sw.Write("\tinput ");
for (int i = 0; i < Inputs.Length; i++)
{
    sw.Write(Inputs[i]);
    if (i != Inputs.Length - 1)
    {
        sw.Write(",");
    }
    sw.Flush();
}
sw.Write(": boolean;\r\n");
if (Outputs.Length > 0)
{
    sw.Write("\toutput ");
    for (int i = 0; i < Outputs.Length; i++)
    {
        sw.Write(Outputs[i]);
        if (i != Outputs.Length - 1)
        {
            sw.Write(",");
        }
        sw.Flush();
    }
    sw.Write(": boolean;\r\n");
}
//sw.Write("\toutput ");
sw.Write("\tVAR ");
for (int i = 0; i < LocalStates.Length; i++)
{
    sw.Write(LocalStates[i]);
    if (i != LocalStates.Length - 1)

```

```

    {
        sw.Write(",");
    }
    sw.Flush();
}
sw.Write(": boolean;\r\n");
sw.Write("ASSIGN\r\n");
for (int i = 0; i < Inputs.Length; i++)
{
    sw.Write("\tinit(" + Inputs[i] + ") :=0;\r\n");
    sw.Flush();
}
for (int i = 0; i < LocalStates.Length; i++)
{
    sw.Write("\tinit(" + LocalStates[i] + ") :=0;\r\n");
    sw.Flush();
}
sw.Write("default\r\n{");
for (int i = 0; i < LocalStates.Length; i++)
{
    sw.Write("\tnext(" + LocalStates[i] + ") :=0;\r\n");
    sw.Flush();
}
for (int i = 0; i < Outputs.Length; i++)
{
    sw.Write("\tnext(" + Outputs[i] + ") :=0;\r\n");
    sw.Flush();
}
sw.Write("}\r\n");
sw.Write("in\r\n{\r\n");
for (int i = 0; i < Rules.Count; i++)
{
    //Rules[i].PrintRule(sw);
    sw.Write("\r\n");
    sw.Flush();
}
sw.Write("}\r\n\r\n");
sw.Write("--Reachability of states\r\n");
for (int i = 0; i < LocalStates.Length; i++)
{
    sw.Write("SPEC EF " + LocalStates[i] + ";\r\n");
    sw.Flush();
}
for (int i = 0; i < Outputs.Length; i++)
{
    sw.Write("SPEC EF " + Outputs[i] + ";\r\n");
    sw.Flush();
}
sw.Write("--Recoverability of states\r\n");
for (int i = 0; i < RecovStates.Length; i++)
{

```

```

        sw.Write("SPEC AG ( " + RecovStates[i] + " -> AF " + RecovStates[i] + "
    );\r\n");
        sw.Flush();
    }
    sw.Write("{}");
    sw.Flush();
    sw.Close();
}
/*public void SaveToVHDL(string Path, bool CreateBus, int OuputSigCount,
OutputTableElement[] OutTable)
{
    String resultCode, tmp;
    int index;
    StringBuilder sb = new StringBuilder();
    StreamWriter sw = new StreamWriter(Path, false);
    FileInfo fi = new FileInfo(Path);
    resultCode = global::sku_to_smv.Properties.Resources.vhd_tmpl;
    //////////////////////////////////////
    index = resultCode.IndexOf("$inputDescription");
    resultCode = resultCode.Remove(index, "$inputDescription".Length);
    if (CreateBus)
    {
        tmp = "-- Входные сигналы\n";
        resultCode = resultCode.Insert(index, tmp);
        index += tmp.Length;
        for (int i = 0; i < Inputs.Length; i++)
        {
            tmp = "-- " + Inputs[i] + "\t->\tinputs[" + i.ToString() + "]\n";
            resultCode = resultCode.Insert(index, tmp);
            index += tmp.Length;
        }
        int j = 0;
        tmp = "-- Выходные сигналы\n";
        resultCode = resultCode.Insert(index, tmp);
        index += tmp.Length;
        for (int i = 0; i < OutTable.Length; i++)
        {
            if (OutTable[i].HasOutput)
            {
                tmp = "-- " + OutTable[i].OutputName + "\t->\toutputs[" + j.ToString() + "]\n";
                resultCode = resultCode.Insert(index, tmp);
                index += tmp.Length;
                ++j;
            }
        }
    }
    //////////////////////////////////////
    while ((index = resultCode.IndexOf("$name")) != -1)
    {
        resultCode = resultCode.Remove(index, "$name".Length);
        resultCode = resultCode.Insert(index, fi.Name.Replace(fi.Extension, ""));
    }
}

```



```

        tmp = "\t\t\t" + OutTable[OutTable.Length - 1].OutputName + "
        :      out          STD_LOGIC\n";
        resultCode = resultCode.Insert(index, tmp);
    }
    else resultCode = resultCode.Remove(index - 2, 1);
}
}
}
////////////////////////////////////
while ((index = resultCode.IndexOf("$width")) != -1)
{
    resultCode = resultCode.Remove(index, "$width".Length);
    resultCode = resultCode.Insert(index, (Rules.Length - 1).ToString());
}
////////////////////////////////////
for (int i = 0; i < Rules.Length; i++)
{
    sb.Append("0");
}
while ((index = resultCode.IndexOf("$zero")) != -1)
{
    resultCode = resultCode.Remove(index, "$zero".Length);
    resultCode = resultCode.Insert(index, sb.ToString());
}
sb.Clear();
////////////////////////////////////
index = resultCode.IndexOf("$localDescription");
resultCode = resultCode.Remove(index, "$localDescription".Length);

tmp = "-- Состояния автомата\n";
resultCode = resultCode.Insert(index, tmp);
index += tmp.Length;
for (int i = 0; i < Rules.Length; i++)
{
    if (Rules[i].output)
    {
        tmp = "-- not used\t->\tcurState[" + i.ToString() + "]\n";
    }
    else tmp = "-- " + Rules[i].Elms[0].Value + "\t->\tcurState[" + i.ToString() +
"]\n";

    resultCode = resultCode.Insert(index, tmp);
    index += tmp.Length;
}
////////////////////////////////////
index = resultCode.IndexOf("$rules");
resultCode = resultCode.Remove(index, "$rules".Length);
for (int i = 0; i < Rules.Length; i++)
{
    if (!Rules[i].output)
    {
        //sb.Append("\tif (");
        sb.Append("\n\t\tnewState(" + i.ToString() + ") := ");
        for (int j = 1; j < Rules[i].Elms.Length; j++)

```

```

        {
            if ((Rules[i].Elems[j].Type != "=") && (Rules[i].Elems[j].Type != "t+1")
&& (!Rules[i].Elems[j].Empty))
            {
                if (Rules[i].Elems[j].Type == "State")
                {
                    if (Rules[i].Elems[j].Inverted)
                    {
                        if (Rules[i].Elems[j].Local)
                        {
                            for (int n = 0; n < Rules.Length; n++)
                            {
                                if (Rules[i].Elems[j].Value == Rules[n].Elems[0].Value)
                                {
                                    //sb.Append("curState(" + n.ToString() + ") = '0'");
                                    sb.Append("(not curState(" + n.ToString() + "))");
                                    break;
                                }
                            }
                        }
                    }
                    else
                    {
                        for (int n = 0; n < Inputs.Length; n++)
                        {
                            if (Inputs[n] == Rules[i].Elems[j].Value)
                            {
                                //
                                //                                     if (CreateBus)
                                //                                     sb.Append("inputs(" +
n.ToString() + ") = '0'");
                                //
                                //                                     else sb.Append(Inputs[n] + "
= '0'");

                                if (CreateBus)
                                    sb.Append("(not inputs(" + n.ToString() + "))");
                                else sb.Append("(not " + Inputs[n] + ")");
                                break;
                            }
                        }
                    }
                }
            }
        }
    }
    else
    {
        if (Rules[i].Elems[j].Local)
        {
            for (int n = 0; n < Rules.Length; n++)
            {
                if (Rules[i].Elems[j].Value == Rules[n].Elems[0].Value)
                {
                    //sb.Append("curState(" + n.ToString() + ") = '1'");
                    sb.Append("curState(" + n.ToString() + ")");
                    break;
                }
            }
        }
    }
}

```

```

    }
    else
    {
        for (int n = 0; n < Inputs.Length; n++)
        {
            if (Inputs[n] == Rules[i].Elems[j].Value)
            {
                // if (CreateBus)
                // sb.Append("inputs(" +
n.ToString() + ") = '1'");
                // else sb.Append(Inputs[n] + "
= '1'");

                if (CreateBus)
                    sb.Append("inputs(" + n.ToString() + ")");
                else sb.Append(Inputs[n]);
                break;
            }
        }
    }
}
else
{
    if (Rules[i].Elems[j].Type == "|") sb.Append(" or ");
    if (Rules[i].Elems[j].Type == "&") sb.Append(" and ");
    if (Rules[i].Elems[j].Type == "(")
    {
        if (Rules[i].Elems[j].Inverted)
        {
            sb.Append(" not ");
        }
        sb.Append("(");
    }
    if (Rules[i].Elems[j].Type == ")")
    {
        sb.Append(")");
    }
}
}
}
int co = 0;
int k = 0;
for (k = 0; k < OutTable.Length; k++)
{
    if (Rules[i].Elems[0].Value == OutTable[k].StateName &&
OutTable[k].HasOutput)
    {
        tmp = OutTable[k].OutputName;
        break;
    }
    else if (OutTable[k].HasOutput)
        co++;
}

```



```

public bool b_Brackets;
state CurrentState;
int currentElement;
String str;

public SympleParser()
{
    b_Brackets = false;
    elements = new SympleElement[0];
    CurrentState = state.START;
    brackets = new SympleElement();
    brackets.StartIndex = -1;
    brackets.EndIndex = -1;
}
static public int FindBracket(String Input, int StartIndex, bool b_UpDown)
{
    int counter = 0;
    if (b_UpDown)
    {
        for (int i = StartIndex; i < Input.Length; i++)
        {
            if (Input[i] == '(')
            {
                counter++;
            }
            if (Input[i] == ')')
            {
                counter--;
            }
            if (counter == 0)
            {
                return i;
            }
        }
    }
    else
    {
        for (int i = StartIndex; i > 0; i--)
        {
            if (Input[i] == ')')
            {
                counter++;
            }
            if (Input[i] == '(')
            {
                counter--;
            }
            if (counter == 0)
            {
                return i;
            }
        }
    }
}

```

```

    }
    return StartIndex;
}
public parceResult Start(String Input)
{
    elements = new SympleElement[0];
    CurrentState = state.START;
    currentElement = -1;
    str = Input.ToLower();

    for (int i = 0; i < str.Length; i++)
    {
        switch (CurrentState)
        {
            case state.START:
                if (str[i] == '#')
                {
                    Array.Resize(ref elements, elements.Length + 1);
                    currentElement++;
                    elements[currentElement] = new SympleElement();
                    elements[currentElement].StartIndex = i;
                    CurrentState = state.COMMENT;
                }
                else if (Regex.IsMatch(str[i].ToString(), "[a-z]"))
                {
                    Array.Resize(ref elements, elements.Length + 1);
                    currentElement++;
                    elements[currentElement] = new SympleElement();
                    elements[currentElement].StartIndex = i;
                    CurrentState = state.SIGNAL;
                }
                else if (str[i] == '(')
                {
                    Array.Resize(ref elements, elements.Length + 1);
                    currentElement++;
                    elements[currentElement] = new SympleElement();
                    elements[currentElement].StartIndex = i;
                    elements[currentElement].EndIndex = i;
                    elements[currentElement].Style = System.Drawing.FontStyle.Bold;
                    CurrentState = state.START;
                }
                else if (str[i] == ')')
                {
                    Array.Resize(ref elements, elements.Length + 1);
                    currentElement++;
                    elements[currentElement] = new SympleElement();
                    elements[currentElement].StartIndex = i;
                    elements[currentElement].EndIndex = i;
                    elements[currentElement].Style = System.Drawing.FontStyle.Bold;
                    CurrentState = state.START;
                }
                else if (str[i] == '[')

```

```

        {
            Array.Resize(ref elements, elements.Length + 1);
            currentElement++;
            elements[currentElement] = new SympleElement();
            elements[currentElement].StartIndex = i;
            CurrentState = state.OPT;
        }
        break;
    case state.COMMENT:
        if (str[i] == '#')
        {
            elements[currentElement].EndIndex = i;
            elements[currentElement].TextColor =
Settings.Default.TextFieldCommentColor;
            elements[currentElement].Style = 0;
            CurrentState = state.START;
        }
        break;
    case state.OPT:
        if (str[i] == ']')
        {
            elements[currentElement].EndIndex = i;
            elements[currentElement].TextColor =
Settings.Default.TextFieldOptionColor;
            elements[currentElement].Style = 0;
            CurrentState = state.START;
        }
        break;
    case state.SIGNAL:
        if (str[i] != ';' && str[i] != '(' && str[i] != ')' && str[i] != '&' && str[i] !=
'" && str[i] != '~' && str[i] != '#' && str[i] != '=')
        {
            break;
        }
        else
        {
            elements[currentElement].EndIndex = i - 1;
            if (str.Substring(elements[currentElement].StartIndex,
elements[currentElement].EndIndex - elements[currentElement].StartIndex + 1) == "t+1")
            {
                elements[currentElement].TextColor = System.Drawing.Color.Red;
                elements[currentElement].Style = 0;
            }
            else
            {
                elements[currentElement].TextColor =
Settings.Default.TextFieldSignalColor;
                elements[currentElement].Style = 0;
            }
            if (str[i] == '#')
            {
                Array.Resize(ref elements, elements.Length + 1);

```



```

        public State endState;
        public Signal signal;
        public Time timeMark;
        public bool SignalInventered;

        public Rule()
        {
            startState = new State();
            endState = new State();
            signal = new Signal();
            timeMark = null;
            SignalInventered = false;
        }

        public Rule(State startState, State endState, Signal signal, Time timeMark, bool
InventeredSignal)
        {
            this.startState = startState;
            this.endState = endState;
            this.signal = signal;
            this.timeMark = timeMark;
            this.SignalInventered = InventeredSignal;
        }

        public DrawArea DrawArea
        {
            get => default;
            set
            {
            }
        }
    }
}

```

Файл «State.cs»

```

using sku_to_smv.src;
using System.Collections.Generic;
using System.Drawing;

namespace sku_to_smv
{
    public enum STATE_TYPE { NONE, INPUT, OUTPUT};
    public class State
    {
        public string Name;      //Имя состояния
        public Point paintDot;
        public Point nameDot;
        public bool Selected;     //если выбрано = true
        public bool Active;
        public List<Signal> inputs;
        public List<Link> links;
        public State()
        {

```

```

        Name = null;
        paintDot = Point.Empty;
        nameDot = Point.Empty;
        Selected = false;
        inputs = new List<Signal>();
        links = new List<Link>();
    }

    public Rule Rule
    {
        get => default;
        set
        {
        }
    }

    public Link Link
    {
        get => default;
        set
        {
        }
    }

    public DrawArea DrawArea
    {
        get => default;
        set
        {
        }
    }

    public override bool Equals(object obj)
    {
        return obj is State state &&
            Name.Equals(state.Name);
    }

    public void setNameDot()
    {
        int radius = Properties.Settings.Default.StateDiametr / 2;
        Font style = Properties.Settings.Default.DefaultText;
        nameDot = new Point(
            paintDot.X + radius - (int)style.Size,
            paintDot.Y + radius - (int)style.Size);
    }

    public void calculateLocation()
    {
        setNameDot();
    }
}

```

```
}
```

Файл «Signal.cs»

```
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Linq;
using System.Text;

namespace sku_to_smv.src
{
    public class Signal
    {
        public String name;
        public Point[] paintDots;
        public List<Point> inventeredPoint;
        public bool Selected;
        public bool Active;
        public Link[] links;

        public Signal()
        {
            name = null;
            paintDots = new Point[0];
            inventeredPoint = new List<Point>();
            Selected = false;
            Active = false;
            links = new Link[0];
        }

        public Signal(string name)
        {
            this.name = name;
        }

        public Rule Rule
        {
            get => default;
            set
            {
            }
        }

        public State State
        {
            get => default;
            set
            {
            }
        }

        public DrawArea DrawArea
        {

```

```

        get => default;
        set
        {
        }
    }

    private void calculatePaintDots()
    {
        paintDots = new Point[links.Length];
        for (int i = 0; i < links.Length; i++)
        {
            Link l = links[i];
            for (int j = 0; j < links[i].signals.Length; j++)
            {
                Signal s = l.signals[j];
                if (s.name.Equals(name))
                {
                    Point p = (l.Arc)
                        ? l.arcDot
                        : new Point(
                            l.startDot.X + (l.endDot.X - l.startDot.X) / 2 + j *
                                (l.startDot.X + (l.endDot.X - l.startDot.X) / 2);
                            l.startDot.Y + (l.endDot.Y - l.startDot.Y) / 2);
                    if (l.invertedSignals.Contains(this))
                    {
                        invertedPoint.Add(p);
                    }
                    paintDots[i] = p;
                }
            }
        }
    }

    public bool isInvert(Point p)
    {
        foreach (Point d in invertedPoint)
        {
            if (d.Equals(p)) return true;
        }
        return false;
    }

    public void calculateLocation()
    {
        calculatePaintDots();
    }
}

```

Файл «Link.cs»

```

using sku_to_smv.src;
using System;
using System.Collections.Generic;

```

```

using System.Drawing;

namespace sku_to_smv
{
    public class Link : State
    {
        public Point startDot;      //точка начала
        public Point endDot;        //точка конца
        public Point arcDot;        //Точка отрисовки арки, если это арка
        public Point[] arrowDots;
        public State startState;    //имя начального состояния
        public State endState;      //имя конечного состояния
        public int lengthLink;
        public int lengthBetweenStatesCentre;
        public string name;
        public bool Arc;            //если арка = true
        public bool Selected;
        public Time timeMark;       //Время перехода
        public Signal[] signals;
        public Signal[] inveteredSignals;
        public float cosx;
        public float sinx;
        public Link()
        {
            startDot = Point.Empty;
            endDot = Point.Empty;
            arcDot = Point.Empty;
            arrowDots = new Point[2];
            startState = null;
            endState = null;
            lengthLink = 0;
            lengthBetweenStatesCentre = 0;
            name = null;
            Arc = false;
            Selected = false;
            timeMark = null;
            signals = new Signal[0];
            inveteredSignals = new Signal[0];
            cosx = 1;
            sinx = 0;
        }

        public Link(State startState, State endState) : this()
        {
            this.startState = startState;
            this.endState = endState;
        }

        public Time Time
        {
            get => default;
            set

```

```

    {
    }
}

public State State
{
    get => default;
    set
    {
    }
}

public DrawArea DrawArea
{
    get => default;
    set
    {
    }
}

public void setName()
{
    this.name = startState.Name + endState.Name;
}

private void setCosX()
{
    if (!Arc) cosx = startState.paintDot.Y == endState.paintDot.Y
        ? (endState.paintDot.X - startState.paintDot.X) / Math.Abs(endState.paintDot.X
- startState.paintDot.X)
        : (float)(endState.paintDot.X - startState.paintDot.X) /
lengthBetweenStatesCentre;
}

private void setSinX()
{
    if (!Arc) sinx = startState.paintDot.X == endState.paintDot.X
        ? (endState.paintDot.Y - startState.paintDot.Y) / Math.Abs(endState.paintDot.Y
- startState.paintDot.Y)
        : (float)(endState.paintDot.Y - startState.paintDot.Y) /
lengthBetweenStatesCentre;
}

private void calculateLength()
{
    int radius = Properties.Settings.Default.StateDiametr / 2;
    lengthBetweenStatesCentre = (int)Math.Sqrt(
        Math.Pow(endState.paintDot.X - startState.paintDot.X, 2) +
        Math.Pow(endState.paintDot.Y - startState.paintDot.Y, 2));
    lengthLink = -radius * 2 + lengthBetweenStatesCentre;
}

```

```

private void caluclateArrowDots()
{
    if (Arc) return;
    int arrowLength = 20; //Длина наконечника стрелки
    float cos30 = (float)Math.Sqrt(3) / 2; //Косинус 30 т.к. угол между линией и
наконечником = 30
    float sin30 = 0.5F; //Синус 30 т.к. угол между линией и наконечником = 30
    float cos1 = cosx * cos30 - sinx * sin30; //cos(a+b) = cos(a)*cos(b) - sin(a)*sin(b);
    float sin1 = sinx * cos30 + cosx * sin30; //sin(a+b) = sin(a)*cos(b) + cos(a)*sin(b);
    float cos2 = cosx * cos30 + sinx * sin30; //cos(a-b) = cos(a)*cos(b) + sin(a)*sin(b);
    float sin2 = sinx * cos30 - cosx * sin30; //sin(a-b) = sin(a)*cos(b) - cos(a)*sin(b);
    arrowDots[0] = new Point((int)(endDot.X - cos1 * arrowLength), (int)(endDot.Y -
sin1 * arrowLength));
    arrowDots[1] = new Point((int)(endDot.X - cos2 * arrowLength), (int)(endDot.Y -
sin2 * arrowLength));
}

private void calculateStartAndEndDots()
{
    if (Arc)
    {
        int arcOffset = 18;
        arcDot = new Point(
            startState.paintDot.X - arcOffset,
            startState.paintDot.Y - arcOffset);
    }
    else
    {
        int radius = Properties.Settings.Default.StateDiametr / 2;
        float x = radius * cosx;
        float y = radius * sinx;
        startDot.X = startState.paintDot.X + radius + (int)x;
        startDot.Y = startState.paintDot.Y + radius + (int)y;
        endDot.X = endState.paintDot.X + radius - (int)x;
        endDot.Y = endState.paintDot.Y + radius - (int)y;
    }
}

public void calculateLocation()
{
    calculateLength();
    setCosX();
    setSinX();
    calculateStartAndEndDots();
    caluclateArrowDots();
}

private bool isInvert(Signal s)
{
    foreach (Signal invS in inveteredSignals)
    {
        if (invS.Equals(s)) return true;
    }
}

```

```

    }
    return false;
}

public bool linkActive()
{
    bool isActive = signals.Length > 0;
    foreach (Signal s in signals)
    {
        bool signalActive = s.Active;
        if (isInvert(s)) signalActive = !signalActive;
        isActive = isActive && signalActive;
    }
    return isActive;
}

public override bool Equals(object obj)
{
    return obj is Link link &&
        name.Equals(link.name);
}
}
}

```

Файл «Time.cs»

```

using System;
using System.Collections.Generic;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace sku_to_smv.src
{
    public class Time : Rule
    {
        public String name;
        public int value;
        public Link[] links;
        public Point[] paintDots;
        public bool selected;

        public Time(String name)
        {
            this.name = name;
            value = 0;
            paintDots = new Point[0];
            links = new Link[0];
            selected = false;
        }

        public TimeTextBox TimeTextBox
        {

```

```

        get => default;
        set
        {
        }
    }

    public DrawArea DrawArea
    {
        get => default;
        set
        {
        }
    }

    private void calculatePainDots()
    {
        paintDots = new Point[links.Length];
        for (int i = 0; i < links.Length; i++)
        {
            Link l = links[i];
            paintDots[i] = new Point(
                (int)(l.startDot.X + (l.endDot.X - l.startDot.X) / 2 + l.signals.Length * 20),
                (int)(l.startDot.Y + (l.endDot.Y - l.startDot.Y) / 2));
        }
    }

    public void calculateLocation()
    {
        calculatePainDots();
    }
}

```

Файл «DrawArea.cs»

```

using System;
using System.CodeDom.Compiler;
using System.Collections.Generic;
using System.Diagnostics;
using System.Drawing;
using System.IO;
using System.IO.Pipes;
using System.Text;
using System.Threading;
using System.Windows.Forms;
using Microsoft.CSharp;
using sku_to_smv.Properties;
using sku_to_smv.src;

namespace sku_to_smv
{
    public class DrawArea : PictureBox
    {

```

```

//States
    public static readonly Pen stateDefaultPen = new
Pen(Settings.Default.ColorDefaultState, 2);
    public static readonly Pen stateSelectedPen = new
Pen(Settings.Default.ColorSelectedState, 3);
    public static readonly Pen stateActivePen = new
Pen(Settings.Default.ColorActiveState, 4);
    public static readonly Pen stateActiveAndSelectedPen = new
Pen(Settings.Default.ColorActiveState, 6);
    public static readonly Brush stateActiveBrush = new
SolidBrush(Settings.Default.ColorFillActiveState);
    public static readonly Font defaultTextFont = Settings.Default.DefaultText;
    public static readonly Brush defaultTextBrush = new
SolidBrush(Settings.Default.ColorDefaultTextBrush);

    public static readonly int StateCentre = Settings.Default.StateCentre;
    public static readonly int OffsetX = Settings.Default.OffsetStateX;
    public static readonly int OffsetY = Settings.Default.OffsetStateY;
    public static readonly int ArcCyrcleradius = Settings.Default.ArcCyrcleradius;
    public static readonly int ArcStartAngle = Settings.Default.ArcStartAngle;
    public static readonly int ArcSweepAngle = Settings.Default.ArcSweepAngle;
    public static readonly int StateDiametr = Settings.Default.StateDiametr;
//Links
    public static readonly Pen linkDefaultPen = new
Pen(Settings.Default.ColorDefaultLink, 2);
    public static readonly Pen linkSelectedPen = new
Pen(Settings.Default.ColorSelectedLink, 4);
//Signals
    public static readonly Font signalDefaultFont = Settings.Default.SignalDefaultFont;
    public static readonly Font signalSelectedFont = Settings.Default.SignalSelectedFont;
    public static readonly Brush signalDefaultBrush = new
SolidBrush(Settings.Default.SignalDefaultColor);
    public static readonly Brush signalActiveBrush = new
SolidBrush(Settings.Default.SignalActiveColor);
//TimeMarks
    public TimeTextBox timeTb;
    private static DrawArea _instnce;

    public static DrawArea getInstance()
    {
        if (_instnce == null)
        {
            _instnce = new DrawArea();
        }
        return _instnce;
    }

    public State[] States;
    public Link[] Links;
    public Signal[] Signals;
    public Time[] TimeMarks;

```

```

public Rule[] Rules;
private float DragOffsetX, DragOffsetY;
private Point PaintDotSelectedState;
public bool SimulStarted;
ToolPanel Tools;

private DrawArea()
{
    InitializeArea();
}
~DrawArea()
{
    Array.Clear(States, 0, States.Length);
    Array.Clear(Links, 0, Links.Length);
    Array.Clear(Signals, 0, Links.Length);
}
private void InitializeArea()
{
    timeTb = TimeTextBox.GetInstance(this);
    Controls.Add(timeTb);
    DragOffsetX = 0;
    DragOffsetY = 0;
    PaintDotSelectedState = Point.Empty;
    SimulStarted = false;
    ((System.ComponentModel.ISupportInitialize)(this)).BeginInit();
    this.SuspendLayout();
    Links = new Link[0];
    States = new State[0];
    Signals = new Signal[0];
    TimeMarks = new Time[0];
    Rules = new Rule[0];
    this.DoubleBuffered = true;
    Tools = ToolPanel.GetInstance();
    Controls.Add(Tools);
    this.MouseClick += new MouseEventHandler(this.mouseClick);
    this.MouseDown += new MouseEventHandler(this.mouseDown);
    this.MouseMove += new MouseEventHandler(this.mouseMove);
    ((System.ComponentModel.ISupportInitialize)(this)).EndInit();
    this.ResumeLayout(false);
}

private void RefreshArea(Graphics g)
{
    g.Clear(Color.White);
    drawStates(g);
    drawLinks(g);
    drawSignals(g);
    drawTimeMarks(g);
}

public void createStates()

```

```

{
    int stateDefaultCentreX = StateCentre;
    int stateDefaultCentreY = StateCentre;
    int offsetStateX = OffsetX;
    int offsetStateY = OffsetY;
    foreach (Rule rule in this.Rules)
    {
        State startState = rule.startState;
        State endState = rule.endState;
        State[] startAndEndStates = { startState, endState };
        foreach (State state in startAndEndStates)
        {
            State s = getStateByName(state.Name);
            if (s == null)
            {
                state.paintDot.X = States.Length % 2 == 0 ? stateDefaultCentreX :
stateDefaultCentreX + offsetStateX;
                state.paintDot.Y = stateDefaultCentreY + States.Length / 2 * offsetStateY;
                addState(state);
            }
        }
    }
}

private void drawStates(Graphics g)
{
    int stateDiametr = StateDiametr;
    foreach (State state in States)
    {
        state.calculateLocation();
        Pen statePen;
        if (!state.Active && !state.Selected) statePen = stateDefaultPen;
        else if (!state.Active && state.Selected) statePen = stateSelectedPen;
        else if (state.Active && !state.Selected) statePen = stateActivePen;
        else statePen = stateActiveAndSelectedPen;
        g.DrawEllipse(statePen, state.paintDot.X, state.paintDot.Y, stateDiametr,
stateDiametr);
        if (state.Active) g.FillEllipse(stateActiveBrush, state.paintDot.X,
state.paintDot.Y, stateDiametr, stateDiametr);
        g.DrawString(state.Name, defaultTextFont, defaultTextBrush, state.nameDot.X,
state.nameDot.Y);
    }
}

private void cancelStates()
{
    States = new State[0];
}

public void createLinks()
{
    foreach (Rule rule in Rules)

```

```

{
    Link link = getLinkByName(rule.startState.Name + rule.endState.Name);
    if (link == null)
    {
        Array.Resize(ref Links, Links.Length + 1);
        link = new Link(rule.startState, rule.endState);
        link.setName();
        link.Arc = link.startState.Equals(link.endState);
        link.startState.links.Add(link);
        link.endState.links.Add(link);
        Links[Links.Length - 1] = link;
    }
    if (rule.SignalInverted)
    {
        Array.Resize(ref link.invertedSignals, link.invertedSignals.Length + 1);
        link.invertedSignals[link.invertedSignals.Length - 1] = rule.signal;
    }
    if (rule.signal != null)
    {
        Array.Resize(ref rule.signal.links, rule.signal.links.Length + 1);
        rule.signal.links[rule.signal.links.Length - 1] = link;
        Array.Resize(ref link.signals, link.signals.Length + 1);
        link.signals[link.signals.Length - 1] = rule.signal;
    }
}
}

private void drawLinks(Graphics g)
{
    int arcRadius = ArcCyrcleradius;
    int arcStartAngle = ArcStartAngle;
    int arcSweepAngle = ArcSweepAngle;
    foreach (Link link in Links)
    {
        Pen linkPen = link.Selected ? linkSelectedPen : linkDefaultPen;
        link.calculateLocation();
        if (link.Arc)
        {
            g.DrawArc(linkPen, new RectangleF(link.arcDot.X, link.arcDot.Y, arcRadius
* 2, arcRadius * 2), arcStartAngle, arcSweepAngle);
        }
        else
        {
            g.DrawLine(linkPen, link.startDot.X, link.startDot.Y, link.endDot.X,
link.endDot.Y);
            g.DrawLine(linkPen, link.endDot.X, link.endDot.Y, link.arrowDots[0].X,
link.arrowDots[0].Y);
            g.DrawLine(linkPen, link.endDot.X, link.endDot.Y, link.arrowDots[1].X,
link.arrowDots[1].Y);
        }
    }
}
}

```

```

private void cancelLinks()
{
    Links = new Link[0];
}

public void createSignals()
{
    foreach (Rule rule in Rules)
    {
        Signal s = getSignalByName(rule.signal.name);
        if (s == null)
        {
            s = rule.signal;
            Array.Resize(ref Signals, Signals.Length + 1);
            Signals[Signals.Length - 1] = s;
        }
    }
}

public void drawSignals(Graphics g)
{
    foreach (Signal s in Signals)
    {
        s.calculateLocation();
        foreach (Point d in s.paintDots)
        {
            Font font = (s.Selected) ? signalSelectedFont : signalDefaultFont;
            bool isActive = s.isInvert(d) ? !s.Active : s.Active;
            Brush brush = (isActive) ? signalActiveBrush : signalDefaultBrush;
            String printedName = (s.invertedPoint.Contains(d)) ? "!" + s.name : s.name;
            g.DrawString(printedName, font, brush, d.X, d.Y);
        }
    }
}

private void cancelSignals()
{
    Signals = new Signal[0];
}

public void createTimeMarks()
{
    foreach (Rule r in Rules)
    {
        if (r.timeMark == null) continue;
        Time timeMark = getTimeMarkByName(r.timeMark.name);
        if (timeMark == null)
        {
            timeMark = r.timeMark;
            Array.Resize(ref TimeMarks, TimeMarks.Length + 1);
            TimeMarks[TimeMarks.Length - 1] = timeMark;
        }
    }
}

```

```

        Link l = getLinkByName(r.startState.Name + r.endState.Name);
        if (l == null) continue;
        Array.Resize(ref timeMark.links, timeMark.links.Length + 1);
        timeMark.links[timeMark.links.Length - 1] = l;
    }
}

public void drawTimeMarks(Graphics g)
{
    foreach (Time t in TimeMarks)
    {
        t.calculateLocation();
        foreach (Point d in t.paintDots)
        {
            Font font = (t.selected) ? signalSelectedFont : signalDefaultFont;
            Brush brush = signalDefaultBrush;
            g.DrawString(t.name + "<" + t.value + ">", font, brush, d.X, d.Y);
        }
    }
}

private void cancelTimeMarks()
{
    TimeMarks = new Time[0];
}

public void cancelDrawElements()
{
    cancelStates();
    cancelLinks();
    cancelSignals();
    cancelTimeMarks();
}

public Link getLinkByName(String name)
{
    foreach (Link link in this.Links)
    {
        if (link.name.Equals(name)) return link;
    }
    return null;
}

private Signal getSignalByName(String name)
{
    foreach (Signal signal in Signals)
    {
        if (signal.name.Equals(name)) return signal;
    }
    return null;
}

private State getStateByName(String name)

```

```

{
    foreach (State state in States)
    {
        if (state.Name.Equals(name)) return state;
    }
    return null;
}

private Time getTimeMarkByName(String name)
{
    foreach(Time t in TimeMarks)
    {
        if (t.name.Equals(name)) return t;
    }
    return null;
}

public State getActiveState()
{
    foreach (State s in States)
    {
        if (s.Active) return s;
    }
    return null;
}

public List<Rule> getRulesWithActiveState()
{
    List<Rule> activeRules = new List<Rule>();
    State activeState = getActiveState();
    foreach(Rule r in Rules)
    {
        if (r.startState.Equals(activeState))
            activeRules.Add(r);
    }
    return activeRules;
}

private void addState(State state)
{
    State currentState = getStateByName(state.Name);
    if (currentState == null)
    {
        Array.Resize(ref States, States.Length + 1);
        States[States.Length - 1] = state;
    }
}

private bool isDotOnLink(Point dot, Link link)
{
    float x = dot.X;
    float y = dot.Y;

```

```

        if (link.Arc)
        {
            int arcRadius = ArcCyrclRadius;
            Point centreArc = new Point(link.arcDot.X + arcRadius, link.arcDot.Y +
arcRadius);
            //Уравнение окружности вида (x-x.centre)^2 + (y-y.centre)^2 = r^2,
представляющей часть арки
            int result = (int)(Math.Pow(x - centreArc.X, 2) + Math.Pow(y - centreArc.Y, 2));
            int squareRadius = (int)Math.Pow(arcRadius, 2);
            bool dotOnPaintedArc = !(x >= centreArc.X && y >= centreArc.Y);
            //Переменная отвечающая, что точка лежит на отрисованной части окружности - арке
            return result >= (squareRadius - 100) && result <= (squareRadius + 100) &&
dotOnPaintedArc;
        }
        else
        {
            //Каноническое уравнение прямой на плоскости типа (x-x1)/(x2-x1) = (y-
y1)/(y2-y1)
            float result = (link.startDot.Y - link.endDot.Y) * x + (link.endDot.X -
link.startDot.X) * y + (link.startDot.X * link.endDot.Y - link.endDot.X * link.startDot.Y);
            int offset = 10;
            float maxX = Math.Max(link.startDot.X, link.endDot.X) + offset;
            float minX = Math.Min(link.startDot.X, link.endDot.X) - offset;
            float maxY = Math.Max(link.startDot.Y, link.endDot.Y) + offset;
            float minY = Math.Min(link.startDot.Y, link.endDot.Y) - offset;
            return (result >= -2500
                && result <= 2500
                && x >= minX && x <= maxX
                && y >= minY && y <= maxY);
        }
    }
}

public bool isDotOnState(Point dot, State state)
{
    int diametr = StateDiametr;
    return dot.X >= state.paintDot.X
        && dot.X <= state.paintDot.X + diametr
        && dot.Y >= state.paintDot.Y
        && dot.Y <= state.paintDot.Y + diametr;
}

public bool isDotOnSignal(Point dot, Signal signal)
{
    float offset = 20;
    bool isOnSignal = false;
    foreach (Point pDot in signal.paintDots)
    {
        isOnSignal = isOnSignal || (dot.X >= pDot.X
            && dot.X <= pDot.X + offset
            && dot.Y >= pDot.Y
            && dot.Y <= pDot.Y + offset);
    }
}

```

```

        return isOnSignal;
    }

    private bool isDotOnTimeMark(Point dot, Time tm)
    {
        int offset = 10 * tm.name.Length;
        bool isOnMark = false;
        foreach (Point pd in tm.paintDots)
        {
            isOnMark = isOnMark
                || dot.X >= pd.X
                && dot.X <= pd.X + offset
                && dot.Y >= pd.Y
                && dot.Y <= pd.Y + offset;
        }
        return isOnMark;
    }

    private bool setSelectedLink(Point dot)
    {
        bool isChanged = false;
        foreach (Link link in Links)
        {
            bool isLink = isDotOnLink(dot, link);
            isChanged = isChanged || link.Selected ^ isLink;
            link.Selected = isLink;
        }
        return isChanged;
    }

    private bool setSelectedState(Point dot)
    {
        bool isChanged = false;
        foreach (State state in States)
        {
            bool isState = isDotOnState(dot, state);
            isChanged = isChanged || state.Selected ^ isState;
            state.Selected = isState;
        }
        return isChanged;
    }

    private bool setActiveState()
    {
        bool isChanged = false;
        foreach (State s in States)
        {
            bool isNeedChangeActivity = s.Selected &&
PaintDotSelectedState.Equals(s.paintDot);
            isChanged = isChanged || isNeedChangeActivity;
            if (isNeedChangeActivity)
            {

```

```

        s.Active = !s.Active;
        cancelStates(s);
    }
}
return isChanged;
}

private void cancelStates(State state)
{
    foreach (State s in States)
    {
        s.Active = s.Equals(state) && s.Active;
    }
}

private bool setSelectedTimeMarks(Point dot)
{
    bool isChanged = false;
    foreach (Time tm in TimeMarks)
    {
        bool isOnMark = isDotOnTimeMark(dot, tm);
        isChanged = isChanged || tm.selected ^ isOnMark;
        tm.selected = isOnMark;
    }
    return isChanged;
}

private bool setActiveSignal()
{
    bool isChanged = false;
    foreach (Signal s in Signals)
    {
        bool newActivityStatus = s.Selected && !s.Active || !s.Selected && s.Active;
        isChanged = isChanged || s.Active ^ newActivityStatus;
        s.Active = newActivityStatus;
    }
    return isChanged;
}

private bool setSelectedSignal(Point dot)
{
    bool isSignalsChanged = false;
    foreach (Signal signal in Signals)
    {
        bool isOnSignal = isDotOnSignal(dot, signal);
        isSignalsChanged = isSignalsChanged || signal.Selected ^ isOnSignal;
        signal.Selected = isOnSignal;
    }
    return isSignalsChanged;
}

private bool setValueTimeMark(Point dot)

```

```

{
    bool tbVisible = false;
    Point tbLocation = Point.Empty;
    Time selectedTm = null;
    foreach (Time tm in TimeMarks)
    {
        tbVisible = tbVisible || tm.selected;
        if (tm.selected)
        {
            tbLocation = dot;
            selectedTm = tm;
        }
    }
    timeTb.Visible = tbVisible;
    timeTb.Location = tbLocation;
    timeTb.timeMark = selectedTm;
    return true;
}

private bool isElementsChanged(Point dot)
{
    bool isLinkChanged = setSelectedLink(dot);
    bool isStateChanged = setSelectedState(dot);
    bool isSignalChanged = setSelectedSignal(dot);
    bool isTimeMarkChanged = setSelectedTimeMarks(dot);
    return isLinkChanged
        || isStateChanged
        || isSignalChanged
        || isTimeMarkChanged;
}

private void relocationStates(Point dot)
{
    foreach (State state in States)
    {
        if (state.Selected)
        {
            state.paintDot.X = dot.X - (int)DragOffsetX;
            state.paintDot.Y = dot.Y - (int)DragOffsetY;
            foreach (Link l in state.links)
            {
                l.calculateLocation();
                if (l.lengthLink < 0 && !l.Arc)
                {
                    int direction = state.Equals(l.startState) ? -1 : 1;
                    state.paintDot.X += (int)(l.cosx * Math.Abs(l.lengthLink) * direction);
                    state.paintDot.Y += (int)(l.sinx * Math.Abs(l.lengthLink) * direction);
                }
            }
        }
    }
}

```

```

private void mouseMove(object sender, MouseEventArgs e)
{
    Point dot = e.Location;
    if (e.Button.Equals(MouseButtons.Left))
    {
        relocationStates(dot);
    }
    if (isElementsChanged(dot) || e.Button.Equals(MouseButtons.Left))
    {
        Refresh();
    }
}

private void mouseDown(object sender, MouseEventArgs e)
{
    foreach (State state in States)
    {
        if (state.Selected)
        {
            DragOffsetX = e.X - state.paintDot.X;
            DragOffsetY = e.Y - state.paintDot.Y;
            PaintDotSelectedState = new Point(state.paintDot.X, state.paintDot.Y);
        }
    }
}

private void mouseClicked(object sender, MouseEventArgs e)
{
    DragOffsetX = 0;
    DragOffsetY = 0;
    bool isStateChangeActivity = !SimulStarted ? setActiveState() : false;
    bool isSignalChangeActivity = setActiveSignal();
    bool s = setValueTimeMark(e.Location);
    if (isStateChangeActivity || isSignalChangeActivity || s) Refresh();
}

protected override void OnPaint(PaintEventArgs e)
{
    base.OnPaint(e);
    e.Graphics.SmoothingMode =
System.Drawing.Drawing2D.SmoothingMode.HighQuality;//Включаем сглаживание
графических объектов
    e.Graphics.TextRenderingHint =
System.Drawing.Text.TextRenderingHint.ClearTypeGridFit;//Включаем сглаживание
шрифтов
    e.Graphics.InterpolationMode =
System.Drawing.Drawing2D.InterpolationMode.HighQualityBicubic;//Включаем
интерполяцию
    RefreshArea(e.Graphics);
}

```

```

public Parser Parser
{
    get => default;
    set
    {
    }
}

public Form1 Form1
{
    get => default;
    set
    {
    }
}
}
}

```

Файл «ToolPanel.cs»

```

using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.Linq;
using System.Text;
using System.Drawing;
using System.Windows.Forms;
using sku_to_smv.Properties;
using sku_to_smv.src;

namespace sku_to_smv
{
    enum MouseState { NONE, CLICK, HOVER };
    enum Alignment { TOP, BOTTOM, LEFT, RIGHT };
    class ToolPanel : Control
    {
        public Orientation PanelOrientation;
        public Alignment PanelAlignment;
        public Collection<Button> Buttons;
        //public DrawArea area;
        public static readonly Brush ToolPanelBrush = new
SolidBrush(SystemColors.Control);
        private static ToolPanel _instance;

        public static ToolPanel getInstance()
        {
            if (_instance == null)
            {
                _instance = new ToolPanel();
            }
            return _instance;
        }

        public delegate void cToolButtonEventHandler(object sender, ToolButtonEventArgs

```

a);

```
public event cToolButtonEventHandler ButtonClicked;

private ToolPanel()
{
    Buttons = new Collection<Button>();
    PanelOrientation = Orientation.Vertical;
    Location = new Point(0, 0);
    Size = new Size(41, 500);
    BackColor = Color.Pink;
    PanelOrientation = Orientation.Vertical;
    PanelAlignment = Alignment.RIGHT;

    Button startButton = new ToolButton();
    //startButton.InactiveImage = Resources.create_simulation;
    startButton.Image = Resources.play;
    startButton.Name = "start";
    startButton.Text = "Запустить симуляцию";
    //startButton.Location = Point.Empty;
    startButton.Click += new EventHandler(onClickSimulStart);
    CalculateLocation(ref startButton);
    Buttons.Add(startButton);

    /*Button runButton = new ToolButton();
    //runButton.SetInactiveImage(Resources.play_grey);
    runButton.Image = Resources.play;
    runButton.Name = "run";
    runButton.Text = "Запуск симуляции";
    CalculateLocation(ref runButton);
    Buttons.Add(runButton);*/

    Button stepButton = new ToolButton();
    //stepButton.SetInactiveImage(Resources.step_grey);
    stepButton.Image = Resources.step;
    stepButton.Name = "step";
    stepButton.Text = "Шаг с остановом";
    CalculateLocation(ref stepButton);
    stepButton.Enabled = false;
    stepButton.Click += new EventHandler(onClickStep);
    Buttons.Add(stepButton);

    Button stopSimulation = new ToolButton();
    //stopSimulation.SetInactiveImage(Resources.stop_grey);
    stopSimulation.Image = Resources.stop;
    stopSimulation.Name = "stop";
    stopSimulation.Text = "Остановить симуляцию";
    CalculateLocation(ref stopSimulation);
    stopSimulation.Enabled = false;
    stopSimulation.Click += onClickStop;
    Buttons.Add(stopSimulation);

    /*Button signalsTable = new ToolButton();
```

```

//signalsTable.SetInactiveImage(Resources.table_grey);
signalsTable.Image = Resources.table;
signalsTable.Name = "table";
signalsTable.Text = "Таблица сигналов";
CalculateLocation(ref signalsTable);
Buttons.Add(signalsTable);

Button cancelSignals = new ToolButton();
//cancelSignals.SetInactiveImage(Resources.table_grey);
cancelSignals.Image = Resources.reset_all;
cancelSignals.Name = "reset";
cancelSignals.Text = "Сбросить все сигналы";
CalculateLocation(ref cancelSignals);
Buttons.Add(cancelSignals);

Button showLog = new ToolButton();
//showLog.SetInactiveImage(Resources.table_grey);
showLog.Image = Resources.show_log;
showLog.Name = "showlog";
showLog.Text = "Показать лог-файл";
CalculateLocation(ref showLog);
Buttons.Add(showLog);*/
Controls.AddRange(Buttons.ToArray());
}

public void CalculateLocation(ref Button button)
{
    int x = 3;
    int y = 3;
    if (PanelOrientation.Equals(Orientation.Vertical))
    {
        y += (button.Size.Height + 5) * Buttons.Count;
    }
    else
    {
        x += (button.Size.Width + 5) * Buttons.Count;
    }
    button.Location = new Point(x, y);
}

/*public void UpdateControlsLocation()
{
    switch (Settings.Default.ToolsPanelLocation)
    {
        case 0:
            PanelOrientation = Orientation.Vertical;
            PanelAlignment = Alignment.LEFT;
            break;
        case 1:
            PanelOrientation = Orientation.Horizontal;
            PanelAlignment = Alignment.TOP;
            break;
        case 2:
            PanelOrientation = Orientation.Vertical;

```

```

        PanelAlignment = Alignment.RIGHT;
        break;
    case 3:
        PanelOrientation = Orientation.Horizontal;
        PanelAlignment = Alignment.BOTTOM;
        break;
    }
}*/

private void onClickSimulStart(object sender, EventArgs e)
{
    DrawArea area = DrawArea.GetInstance();
    State activeState = area.getActiveState();
    if (activeState == null)
    {
        MessageDialog.activeStateIsNull();
        return;
    }
    area.SimulStarted = true;
    Buttons[0].Enabled = false;
    Buttons[1].Enabled = true;
    Buttons[2].Enabled = true;
}

private void onClickStep(object sender, EventArgs e)
{
    DrawArea area = DrawArea.GetInstance();
    State activeState = area.getActiveState();
    if (activeState == null) return;
    List<Rule> activeRules = area.getRulesWithActiveState();
    bool anythingTransferred = false;
    bool transferredArc = false;
    foreach (Rule r in activeRules)
    {
        Link l = area.getLinkByName(r.startState.Name + r.endState.Name);
        if (l == null) return;
        bool linkActive = l.linkActive();
        r.endState.Active = linkActive;
        transferredArc = l.Arc && linkActive;
        anythingTransferred = anythingTransferred || linkActive;
    }
    activeState.Active = !anythingTransferred || transferredArc;
    area.Refresh();
}

private void onClickStop(object sender, EventArgs e)
{
    DrawArea area = DrawArea.GetInstance();
    area.SimulStarted = false;
    Buttons[0].Enabled = true;
    Buttons[1].Enabled = false;
    Buttons[2].Enabled = false;
}

```

```

    }
}

```

```

public class ToolButtonEventArgs : EventArgs
{
    public ToolButtonEventArgs(string s)
    {
        ButtonName = s;
    }
    private string ButtonName;
    public string Name
    {
        get { return ButtonName; }
    }
}

```

```

class ToolButton : Button
{
    Bitmap image;
    Bitmap imageInactive;
    Bitmap frame;
    Bitmap frame_hover;
    Bitmap frame_click;
    MouseState ms;

    public ToolButton()
    {
        image = null;
        imageInactive = null;
        frame = null;
        frame_hover = null;
        frame_click = null;
        Location = Point.Empty;
        Size = new Size(35, 35);
        BackColor = SystemColors.Control;
        Name = "Button";
        Text = "toolButton";
        Enabled = true;
        Visible = true;
        ms = MouseState.NONE;
        BackColor = Color.FromArgb(0, 0, 0, 0);
        SetFrame(Resources.frame);
        SetFrameHover(Resources.frame_hover);
        SetFrameClick(Resources.frame_click);
    }
}

```

```

public Form1 Form1
{
    get => default;
    set
    {

```

```

    }
}

public void Draw(Graphics g)
{
    g.Clear(Color.White);
    if (Visible)
    {
        //g.FillRectangle(new SolidBrush(BackColor), new Rectangle(Location, Size));
        if (true)
        {
            if (Image != null)

                g.DrawImage(Image, new Rectangle(new Point(Location.X + 4,
Location.Y + 4), new Size(22, 22)));
            /*switch (ms)
            {
                case MouseState.NONE:
                    if (frame != null)
                        g.DrawImage(frame, new Rectangle(Location, Size));
                    break;
                case MouseState.HOVER:
                    if (frame_hover != null)
                        g.DrawImage(frame_hover, new Rectangle(Location, Size));
                    break;
                case MouseState.CLICK:
                    if (frame_click != null)
                        g.DrawImage(frame_click, new Rectangle(Location, Size));
                    break;
            }*/
        }
        else
        {
            /*if (imageInactive != null)
                g.DrawImage(imageInactive, new Rectangle(new Point(Location.X + 4,
Location.Y + 4), new Size(22, 22)));
            if (frame != null)
                g.DrawImage(frame, new Rectangle(Location, Size));*/
        }
    }
}

public void SetImage(String path)
{
    try
    {
        image = new Bitmap(path);
        image.MakeTransparent(System.Drawing.Color.Magenta);
    }
    catch (System.IO.FileNotFoundException)
    {
        image = null;
    }
}

```

```

    }
    public void SetImage(Bitmap im)
    {
        image = im;
        image.MakeTransparent(System.Drawing.Color.Magenta);
    }
    public void SetInactiveImage(String path)
    {
        try
        {
            imageInactive = new Bitmap(path);
            imageInactive.MakeTransparent(System.Drawing.Color.Magenta);
        }
        catch (System.IO.FileNotFoundException)
        {
            imageInactive = null;
        }
    }
    public void SetInactiveImage(Bitmap im)
    {
        imageInactive = im;
        imageInactive.MakeTransparent(System.Drawing.Color.Magenta);
    }
    public void SetFrame(String path)
    {
        try
        {
            frame = new Bitmap(path);
            frame.MakeTransparent(System.Drawing.Color.Magenta);
        }
        catch (System.IO.FileNotFoundException)
        {
            frame = null;
        }
    }
    public void SetFrame(Bitmap im)
    {
        frame = im;
        frame.MakeTransparent(System.Drawing.Color.Magenta);
    }
    public void SetFrameHover(String path)
    {
        try
        {
            frame_hover = new Bitmap(path);
            frame_hover.MakeTransparent(System.Drawing.Color.Magenta);
        }
        catch (System.IO.FileNotFoundException)
        {
            frame_hover = null;
        }
    }
}

```

```

public void SetFrameHover(Bitmap im)
{
    frame_hover = im;
    frame_hover.MakeTransparent(System.Drawing.Color.Magenta);
}
public void SetFrameClick(String path)
{
    try
    {
        frame_click = new Bitmap(path);
        frame_click.MakeTransparent(System.Drawing.Color.Magenta);
    }
    catch (System.IO.FileNotFoundException)
    {
        frame_click = null;
    }
}
public void SetFrameClick(Bitmap im)
{
    frame_click = im;
    frame_click.MakeTransparent(System.Drawing.Color.Magenta);
}
public MouseState CheckMouseState(MouseEventArgs e)
{
    if (e != null)
        if ((e.X > Location.X && e.X < Location.X + Size.Width && e.Y > Location.Y
&& e.Y < Location.Y + Size.Height) && Visible)
        {
            if (e.Button == MouseButton.Left && Enabled)
            {
                return ms = MouseState.CLICK;
            }
            return ms = MouseState.HOVER;
        }
        return ms = MouseState.NONE;
    }
}
}

```

Файл “Signal.panel”

```

using sku_to_smv.Properties;
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace sku_to_smv.src.components
{
    class SignalPanel : Control
    {

```

```

private static SignalPanel _instance;

public static SignalPanel getInstance()
{
    if (_instance == null)
    {
        _instance = new SignalPanel();
    }
    return _instance;
}

public static void updateInstance()
{
    _instance = new SignalPanel();
}

DrawArea da = DrawArea.getInstance();
public Orientation PanelOrientation;
public Alignment PanelAlignment;
public static readonly Font signalDefaultFont = Settings.Default.SignalDefaultFont;
public static readonly Font signalSelectedFont = Settings.Default.SignalSelectedFont;

private SignalPanel()
{
    Location = new Point(da.Size.Width - 41, 0);
    Size = new Size(41, 500);
    BackColor = Color.Pink;
    PanelOrientation = Orientation.Vertical;
    PanelAlignment = Alignment.RIGHT;
    updateButtons();
}

private void updateButtons()
{
    for (int i = 0; i < da.Signals.Length; i++)
    {
        Signal s = da.Signals[i];
        Button b = new Button();
        b.Name = s.name;
        b.Text = s.name;
        b.Click += new EventHandler(onButtonAction);
        b.Size = new Size(35, 35);
        b.BackColor = Color.FromArgb(0, 0, 0, 0);
        b.Enabled = true;
        b.Visible = true;
        CalculateLocation(ref b, i);
        Controls.Add(b);
    }
}

private void onButtonAction(object sender, EventArgs e)
{

```

```

        Button b = (Button) sender;
        Signal s = da.getSignalByName(b.Name);
        s.Active = !s.Active;
        b.ForeColor = s.Active ? Color.Red : Color.Black;
        b.Refresh();
        da.Refresh();
    }

    public void CalculateLocation(ref Button button, int count)
    {
        int x = 3;
        int y = 3;
        if (PanelOrientation.Equals(Orientation.Vertical))
        {
            y += (button.Size.Height + 5) * count;
        }
        else
        {
            x += (button.Size.Width + 5) * count;
        }
        button.Location = new Point(x, y);
    }

    public override void Refresh()
    {
        Location = new Point(da.Size.Width - 41, 0);
        base.Refresh();
    }
}
}

```

ПРИЛОЖЕНИЕ Б

Листинги приложения авторизации

Файл application.yaml

```
server:
  port: 8081
spring:
  mvc:
    view:
      prefix: /templates/
      suffix: .html
  keycloak:
    auth-server-url: 'http://localhost:8484/auth'
    realm: realm-app
    resource: keycloak-app
    credentials:
      secret: hDfwKeZ39unR4qoiRzzj5p0tYtZlXraq
    security-constraints:
      - authRoles:
          user
        securityCollections:
          - patterns:
              /user
      - authRoles:
          admin
        securityCollections:
          - patterns:
              /admin
      - authRoles:
          user
        securityCollections:
          - patterns:
              /api
```

Класс WebMvcConfiguration

```
package com.company.controller;

import org.springframework.boot.web.server.ErrorPage;
import org.springframework.boot.web.server.WebServerFactoryCustomizer;
import org.springframework.boot.web.servlet.server.ConfigurableServletWebServerFactory;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.http.HttpStatus;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;

@Configuration
public class WebMvc implements WebMvcConfigurer {

    @Bean
    public WebServerFactoryCustomizer<ConfigurableServletWebServerFactory>
    containerCustomizer() {
        return container ->
```

```

        container.addErrorPages(new
            ErrorPage(HttpStatus.FORBIDDEN,
                "/denied"));
    };
}
}

```

Класс KeycloakMVCController.

```

package
    com.company.controller;

import
    org.keycloak.KeycloakPrincipal;
import
    org.springframework.stereotype.Controller;
import
    org.springframework.ui.Model;
import
    org.springframework.web.bind.annotation.GetMapping;
import
    org.springframework.web.bind.annotation.RequestMapping;

@Controller
@RequestMapping
public
    class
        KeycloakMVCController
    {
        @GetMapping("/user")
        public
            String
            user(KeycloakPrincipal
                principal,
                Model
                m)
            {
                String
                    username
                =
                principal.getKeycloakSecurityContext().getToken().getPreferredUsername();
                m.addAttribute("name",
                    username);
                return
                    "access";
            }

        @GetMapping("/admin")
        public
            String
            admin(KeycloakPrincipal
                principal,
                Model
                m)
            {
                String
                    username
                =
                principal.getKeycloakSecurityContext().getToken().getPreferredUsername();
                m.addAttribute("name",
                    username);
                return
                    "access";
            }

        @GetMapping("/denied")
        public
            String
            denied(KeycloakPrincipal
                principal,
                Model
                m)
            {
                String
                    username
                =
                principal.getKeycloakSecurityContext().getToken().getPreferredUsername();
                m.addAttribute("name",
                    username);
                return
                    "no_access";
            }
    }
}

```

Класс KeycloakRestController.

```

package
    com.company.controller;

import
    org.springframework.web.bind.annotation.GetMapping;
import
    org.springframework.web.bind.annotation.RequestMapping;
import
    org.springframework.web.bind.annotation.RestController;

@RestController
@RequestMapping("/api")
public
    class
        KeycloakRestController
    {

```

```

    @GetMapping
    public String successResponseOn /api path";
    }
}

Класс KeycloakAppApplication.
package com.company;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.boot.autoconfigure.security.servlet.SecurityAutoConfiguration;

@SpringBootApplication(exclude = SecurityAutoConfiguration.class)
public class KeycloakAppApplication {

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

```

Файл access.html

```

<!DOCTYPE html>
<html xmlns:th="https://www.thymeleaf.org"
>
<head>

<style>
div {
    background: #5EFE3E; /* Цвет фона */
    color: #000000; /* Цвет текста */
    padding: 8%; /* Поля вокруг текста */
    border-radius: 15px; /* Уголки */

    margin-left: 10%;
    margin-right: 10%;
    margin-top: 10%;
    margin-bottom: 10%;
}
</style>
</head>
<body bgcolor="#24F500">

<div align="center"><h1>Доступ разрешен пользователю <span th:text="${name}"></span>
</h1></div>
</body>
</html>

```

Файл denied.html

```

<!DOCTYPE html>
<html xmlns:th="https://www.thymeleaf.org"
<head>

<style>
div {

```

```

background:      #FF6342;          /*      Цвет      фона      */
color:           #000000;          /*      Цвет      текста      */
padding:         8%;               /*      Поля      вокруг      текста      */
border-radius:   15px;             /*      Уголки      */

margin-left:                                           10%;
margin-right:                                          10%;
margin-top:                                           10%;
margin-bottom:                                        10%;
}
</style>
</head>
<body
                                bgcolor="#FF4018">

<div      align="center">      <h1>Доступ      запрещен      пользователю      <span
th:text="{name}"></span></h1></div>
</body>
</html>

```

ПРИЛОЖЕНИЕ В

Презентация

Система темпорального анализа моделей параллельных вычислительных систем

Автор работы: студент группы 20ВВм1 Борискин Вячеслав
Владимирович

Научный руководитель: к.т.н., доцент Синев Михаил Петрович

Цель и объект работы

Цели работы:

- исследовать методы моделирования распределенных ИС,
- проанализировать способы защиты информации в распределенных ИС.

Задачи работы:

- разработка среды моделирования ИС посредством конечных автоматов,
- моделирование в разработанной среде алгоритма защиты данных, используемых в распределенных ИС,
- разработка программы, реализующей исследуемый алгоритм защиты данных в распределенных ИС.

HyperText Transport Protocol Secure (HTTPS)

HTTPS - защищенный, базовый протокол при передаче данных между отправителем и получателем по сети. Основывается на использовании TLS-сертификатов и использует следующие алгоритмы.

- RSA,
- Диффи-Хеллмана.

3

RSA (Rivest, Shamir, Adleman)

- Имеются публичный и приватный ключи,
- Информация зашифровывается только публичным ключом и передается в зашифрованном виде, а расшифровывается только приватным,
- Ключи не взаимозаменяемые .

4

Диффи-Хеллмана

- Основное свойство алгоритма, в том что информация зашифровывается и расшифровывается одним и тем ключом.
- Данный алгоритм работает намного быстрее асимметричного RSA, но менее безопасный.

5

Установка https-соединения



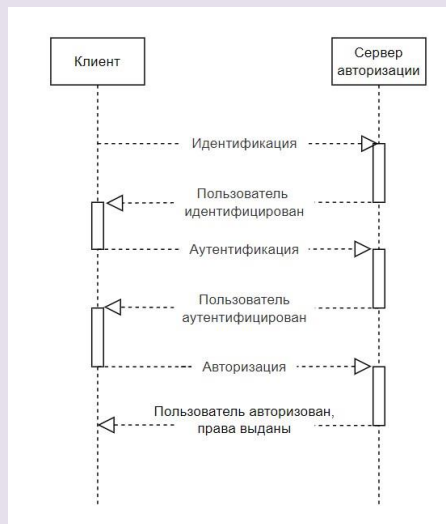
6

Защита данных

- Идентификация. Выявление и установка пользователя по его идентификатору (имя, логин, электронная почта, номер телефона т. д.). Цель данной фазы определить, с каким именно пользователем будут происходить дальнейшие проверки,
- Аутентификация. Цель фазы - удостоверение, что идентифицируемый пользователь является именно тем, за кого себя выдает. Например, проверка пользователя по паролю, электронной подписи, сертификату и т.д,
- Авторизация. Установка доступа аутентифицированного пользователя к защищенному ресурсу. Цель фазы - выдача пользователю, прошедшему аутентификацию прав на действия с запрашиваемым ресурсом.

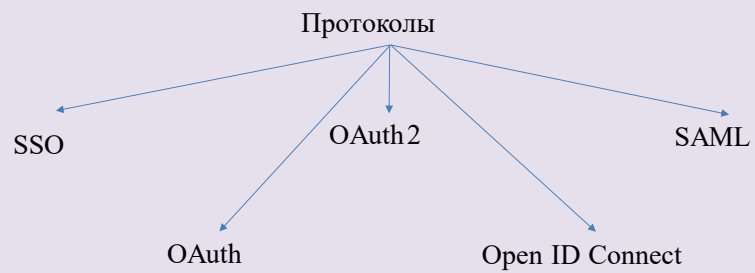
7

Доступ к защищенным данным



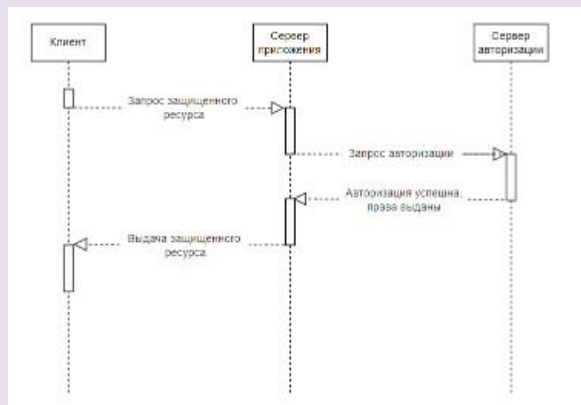
8

Протоколы безопасности



9

Протокол SSO



10

[illegible]

Протокол Open Id Connect

```
{  
  "alg": "HS256",  
  "typ": "JWT"  
}
```

```
{  
  "sub": "1234567890",  
  "name": "John Doe",  
  "admin": true  
}
```

```
HMACSHA256(  
  base64UrlEncode(header) + "." +  
  base64UrlEncode(payload),  
  secret)
```

12

Автоматы

Абстрактные автоматы – вычислительные машины, представленные в виде математических моделей, осуществляющие обработку информации по заданному алгоритму

Типы автоматов

```
graph TD; A[Типы автоматов] --> B[Детерминированные]; A --> C[Недетерминированные]
```

Детерминированные

Недетерминированные

13

Существующие методики моделирования

Цели Маркова

Сети Петри

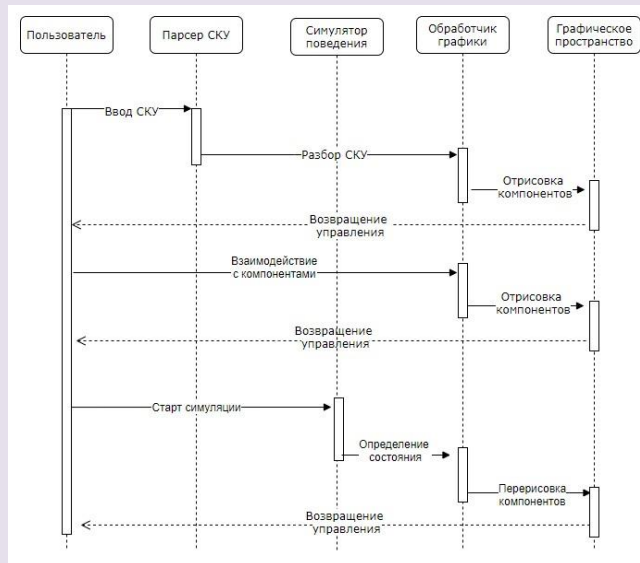
Конечные автоматы

14

The diagram shows a Markov chain with two states, S_0 and S_1 . The transitions are labeled as follows:

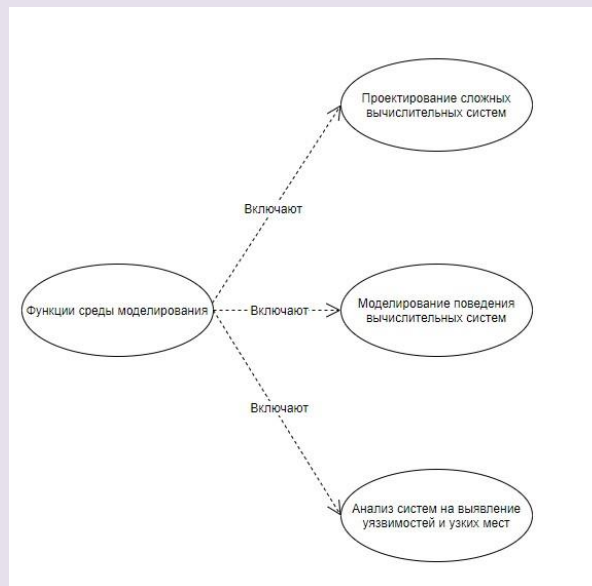
- From S_0 to S_0 : Z_0/W_0
- From S_0 to S_1 : Z_0/W_1
- From S_1 to S_0 : Z_1/W_1
- From S_1 to S_1 : Z_1/W_0

UML-диаграмма последовательности



15

UML-диаграмма вариантов использования



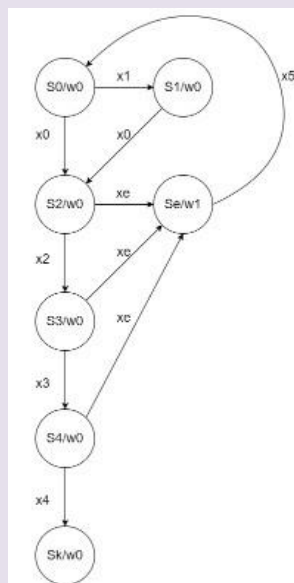
16

Стек технологий



17

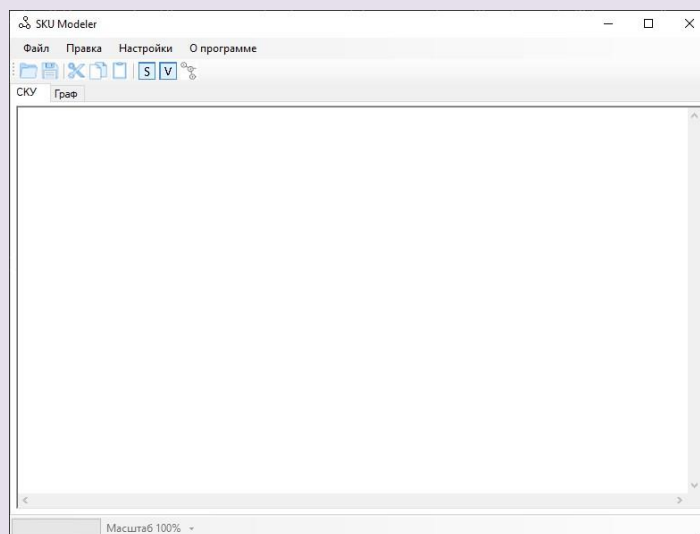
Конечный автомат



S0	Начальное состояние
S0	Исходное состояние
S1	Авторизация на сервере
S2	Предоставление доступа
S3	Запрос кода авторизации
S4	Обмена кода авторизации на токен
Sk	Конечное состояние Пользователь авторизован
Se	Ошибочное состояние
X0	Переход на страницу выдачи прав
X1	Переход на страницу входа
X2	Права выданы
X3	Кода авторизации получен
X4	Токен получен
X5	Переход в исходное состояние
Xc	Ошибка
W0	Успех
W1	Ошибка

18

Текстовое пространство



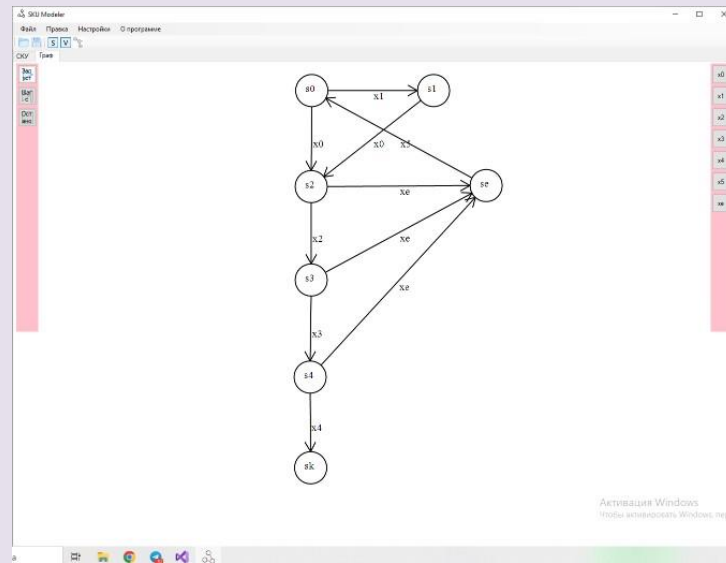
19

Ввод СКУ

```
СКУ    Граф
input=[x0,x1,x2,x3,x4,x5,x6];
s0=se&x5;
s1=s0&x1;
s2=s0&x0;
s2=s1&x0;
s3=s2&x2;
s4=s3&x3;
sk=s4&x4;
se=s2&x6;
se=s3&x6;
se=s4&x6;
```

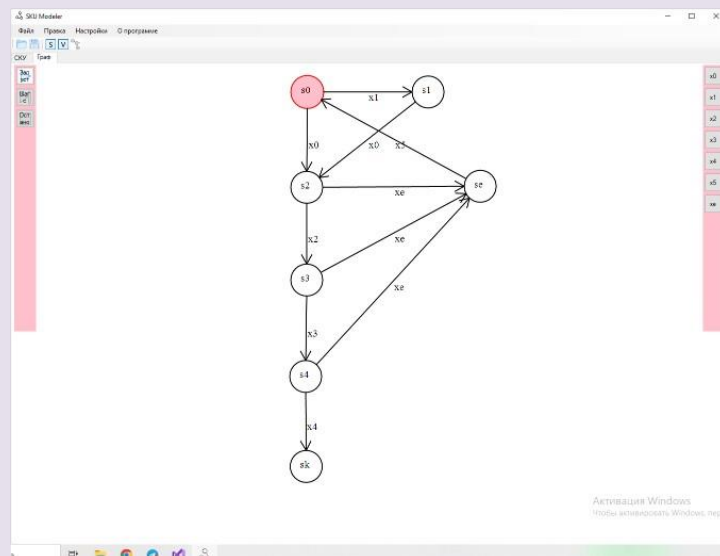
20

Конечный автомат



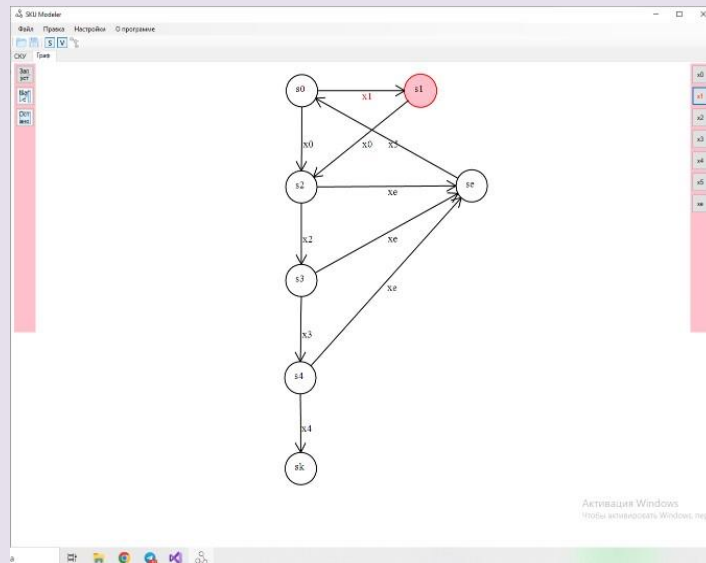
21

Моделирование КА



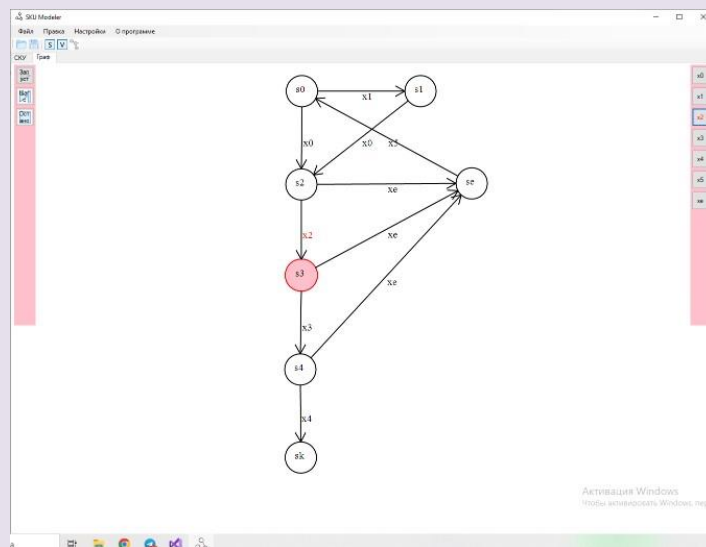
22

Моделирование КА



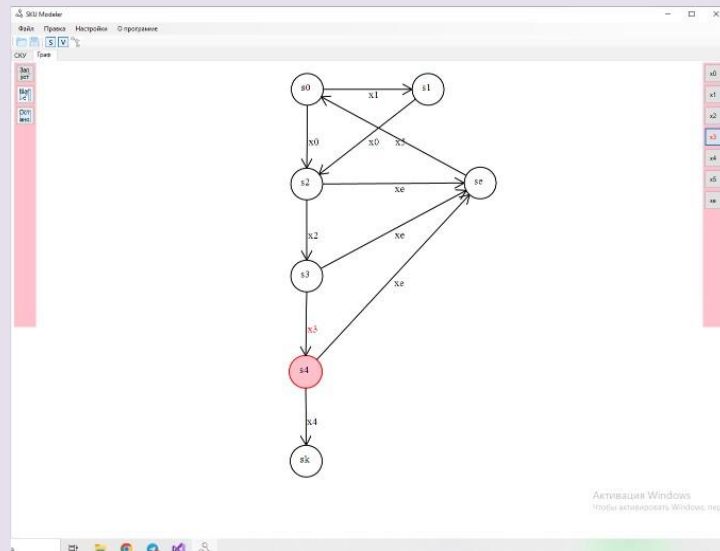
23

Моделирование КА



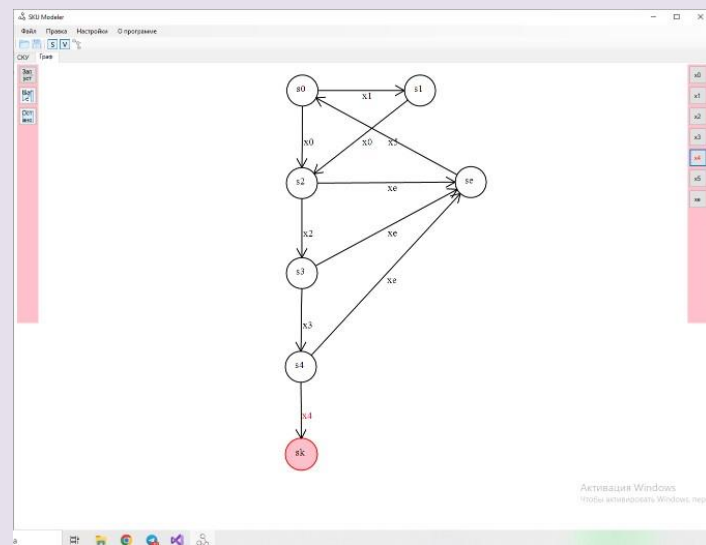
24

Моделирование КА



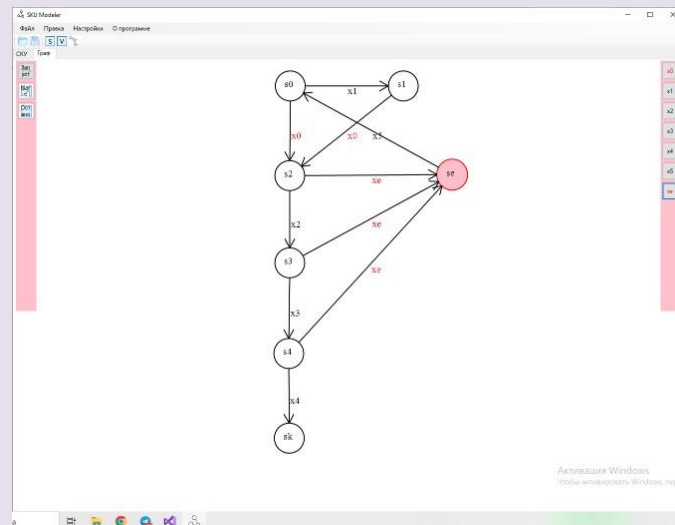
25

Моделирование КА



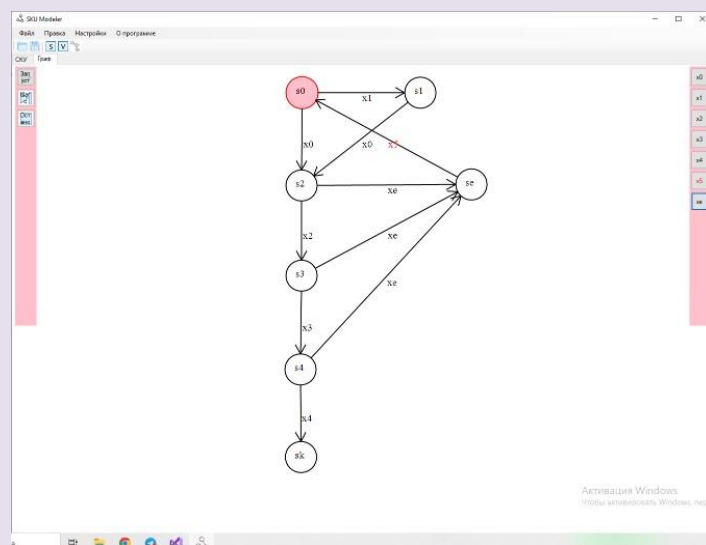
26

Моделирование КА



27

Моделирование КА



28

KeyCloak

Keycloak является SSO (Single Sign On) решением для собственной реализации протоколов OAuth2 и OpenID Connect (OIDC) в разрабатываемых приложениях.

29

Возможности KeyCloak

- Single -Sign On and Single -Sign Out для браузерных приложений.
- Поддержка OpenID/OAuth 2.0/SAML.
- Identity Brokering – аутентификация с помощью внешних OpenID Connect или SAML идентификационных провайдеров.
- Social Login – поддержка Google, GitHub, Facebook, Twitter для идентификации пользователей.
- User Federation – синхронизация пользователей из LDAP и Active Directory серверов и других идентификационных провайдеров.
- Admin Console — для единого управления настройками и параметрами решения через Web.
- Клиентские адаптеры для JavaScript applications, WildFly, JBoss EAP, Fuse, Tomcat, Jetty, Spring.

30

Создание Realm

The screenshot shows the 'Create New Realm' form in the Keycloak administration console. The left sidebar is titled 'Realm-app' and contains a 'Configure' section with 'Realm Settings' selected. The main form has tabs for 'General', 'Login', 'Keys', 'Email', 'Themes', 'Localization', 'Cache', 'Tokens', 'Client Registration', 'Client Policies', and 'Security Defenses'. The 'General' tab is active, showing fields for 'Name' (realm-app), 'Display name', 'HTML Display name', and 'Frontend URL'. There are toggle switches for 'Enabled' (ON) and 'User-Managed Access' (OFF). An 'Endpoints' section shows 'OpenID Endpoint Configuration' with a value of 'SAML 2.0 Identity Provider Metadata'. 'Save' and 'Cancel' buttons are at the bottom.

31

Создание клиента

The screenshot shows the 'Create New Client' form in the Keycloak administration console. The left sidebar is titled 'Keycloak-app' and contains a 'Settings' section with 'Client Settings' selected. The main form has tabs for 'Settings', 'Credentials', 'Keys', 'Roles', 'Client Scopes', 'Mappers', 'Scope', 'Authorization', 'Revocation', 'Sessions', and 'Offline Access'. The 'Settings' tab is active, showing fields for 'Client ID' (keycloak-app), 'Name', and 'Description'. There are toggle switches for 'Enabled' (ON), 'Always Display In Console' (OFF), 'Consent Required' (OFF), 'Standard Flow Enabled' (ON), 'Implicit Flow Enabled' (OFF), 'Direct Access Grants Enabled' (ON), 'Service Accounts Enabled' (ON), 'OAuth 2.0 Device Authorization Grants Enabled' (OFF), 'OIDC OPA Grants Enabled' (OFF), 'Authorization Enabled' (ON), and 'Front Channel Logout' (OFF). There are dropdown menus for 'Login Theme', 'Client Protocol' (openid-connect), and 'Access Type' (confidential). A 'Root URL' field is present, and a 'Valid Redirect URIs' field contains 'http://localhost:8081/*'. A 'Base URL' field is at the bottom.

32

Создание пользователей

Add user

ID

Created At

Username *

Email

First Name

Last Name

User Enabled ☒

Email Verified ☐

Groups

Required User Actions

Add user

ID

Created At

Username *

Email

First Name

Last Name

User Enabled ☒

Email Verified ☐

Groups

Required User Actions

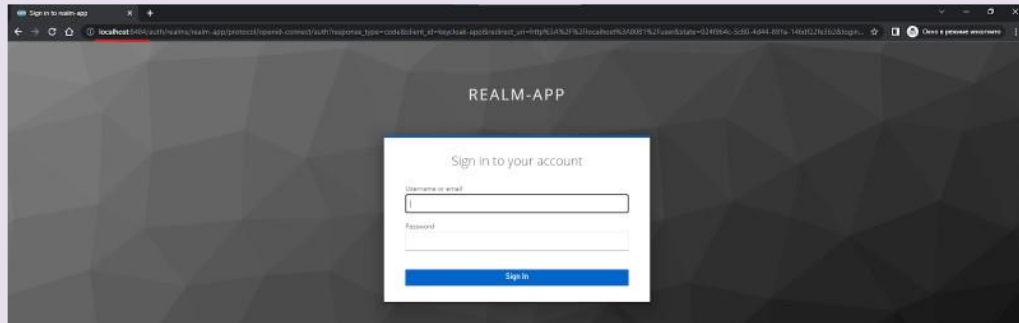
33

Разработка клиентского приложения

```
1 server:
2   port: 8081
3 spring:
4   mvc:
5     view:
6       prefix: /templates/
7       suffix: .html
8 keycloak:
9   auth-server-url: 'http://localhost:8484/auth'
10  realm: realm-app
11  resource: keycloak-app
12  credentials:
13    secret: hDfwKeZ39unR4qoiRzzj5p0tYtZlxraq
14  security-constraints:
15    - authRoles:
16      - user
17      securityCollections:
18        - patterns:
19          - /user
20    - authRoles:
21      - admin
22      securityCollections:
23        - patterns:
24          - /admin
```

34

Страница авторизации в Keycloak



35

Доступ разрешен



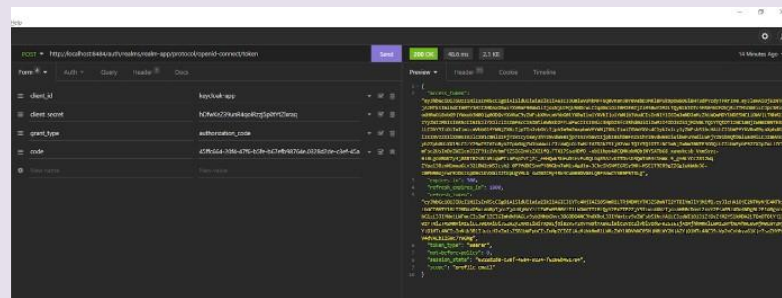
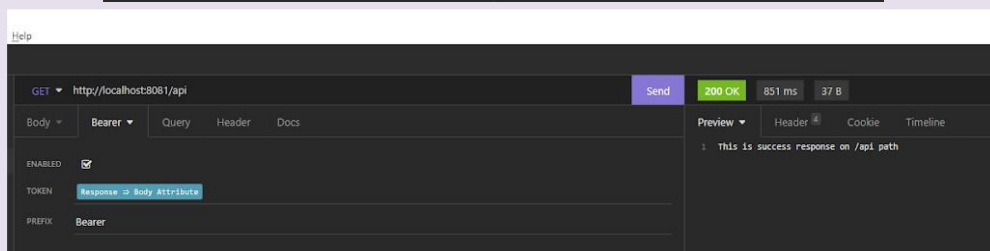
36

Доступ запрещен



37

Доступ к ресурсу по API



38

Результаты работы

- Разработано и отлажено приложение, выполняющее задачи проектирования анализа и моделирования поведения распределённых информационных систем,
- Разработано и отлажено алгоритм разбора SKU, описывающей моделируемую ИС,
- Разработано и отлажено алгоритм отображения графической модели ВС, представленной в виде конечного автомата,
- Разработано и отлажено алгоритм симуляции работы ВС,
- Исследованы протоколы защиты данных OAuth2 и OIDC,
- Разработано приложение, реализующее исследованные протоколы,
- Приложение было представлено на семинаре по электронным и сетевым технологиям MWENT-2020. Результаты трудов семинара проиндексированы рефератной базой данных SCOPUS.

39

Спасибо за внимание!

40