СОДЕРЖАНИЕ

Введение 5
1 Анализ прототипов, литературных источников и формирование требований к проектирумому программному средству 7
1.1 Анализ существующих прототипов 7
1.2 Постановка задачи 11
2 Анализ требований к программному средству и разработка функциональных требований 12
2.1 Описание функциональности программного средства 12
2.2 Спецификация функциональных требований 13
3 Проектирование программного средства 14
3.1 Модуль авторизации и аутентификации 14
3.2 Модуль REST Api 16
3.3 Модуль парсинга 18
4 Создание программного средства 19
5 Тестирование, проверка работоспособности и анализ полученных результатов 22
5.1 Тестирование, проверка работоспособности и анализ полученных результатов 22
6 Руководство по установке и использованию 28
6.1 Установка программного средства 28
6.2 Руководство пользователя 29
Заключение 35
Список использованных источников 36
Приложение А. Исходный код программы 37
Приложение Б. Исходный код парсера 73
Приложение В 95

ВВЕДЕНИЕ

Веб-сервис (или веб-служба) – идентифицируемая уникальным веб-адресом программная система со стандартизированными интерфейсами, а также HTML-документ сайта, отображаемый браузером пользователя. Веб-службы взаимодействуют с пользователем посредством графического интерфейса HTML документа или сообщений, основанных на определенных протоколах и соглашениях. В обиходе веб-сервисами называют различные услуги, оказываемые в Интернете.

Наиболее популярными примерами веб-служб являются поисковые системы, которые предоставляют пользователям возможность искать необходимую информацию в Интернете, посредством автоматического разбора HTML документов. Известные всем стриминговые сервисы (услуги по выдаче потокового мультимедиа), на подобие Spotify, Apple Music, Netflix, Megogo и т.д., также являются веб-службами.

В преобладающем большинстве случаев интерфейс взаимодействия с сервисами не ограничивается только лишь HTML документами, отображаемых браузерами, а также включает в себя различные протоколы взаимодействия. Это, чаще всего, обосновано разграничением ответственности между программными средствами, отвечающих за взаимодействие с пользователем и выполнение конкретной логики, что позволяет более гибко настроить систему и, при необходимости, ее расширить. Примером хорошего разграничения можно взять ресурс Random.org, предоставляющий пользователям интерфейс для генерации «настоящих случайных чисел». В данном сервисе можно выделить как минимум три уровня – веб-сайт (графический интерфейс), rest api (программный интерфейс) и сам генератор чисел. Так как такой подход является более гибким при разработке, было сделано решение вести данный курсовой проект по этому примеру.

REST (от англ. Representational State Transfer — «передача состояния представления») — архитектурный стиль взаимодействия компонентов распределённого приложения в сети. REST представляет собой согласованный набор ограничений, учитываемых при проектировании ПС. Основным таким ограничением является отсутствие состояния, т.е. в каждом запросе от пользователя (будь то иное ПС или сам веб-сервис) содержится вся необходимая информация для работы службы. Для веб-служб, построенных с учетом этих ограничений применяется термин RESTful.

Одним из наиболее главных требований к REST службе является единообразие интерфейса – четко установленные «точки» взаимодействия с веб-сервисом позволяет разработчикам максимально гибко взаимодействовать с основным ПС (к которому конечный пользователь желает получить доступ), при этом полностью абстрагируя пользователя от деталей реализации, т.е. обращаясь с этой службе клиент не способен точно

определить, с чем он взаимодействует. Таким образом разработчик всегда может разграничить доступ к сервису, тем самым повысив производительность всей системы, особенно если предоставляемая услуга является ресурсозатратной.

Такая архитектура была выбрана в данном курсовом проекте, т.к. основной предоставляемой услугой стал синтаксический разбор языка C, что может сильно нагрузить систему, при умеренном количестве запросов. Синтаксический анализ (или разбор) в лингвистике и информатике — процесс сопоставления линейной последовательности лексем (слов, токенов) естественного или формального языка с его формальной грамматикой. Результатом обычно является дерево разбора (синтаксическое дерево). Обычно применяется совместно с лексическим анализом.

Основной целью синтаксического разбора является последующее использование дерева разбора для семантического анализа и трансляции программы, однако в данной работе синтаксический анализ (и конечное абстрактное синтаксическое дерево) были направлены для последующего построения блок-схемы написанного кода.

Данная работа состоит из двух основных частей: синтаксический анализатор и веб-служба. Для создания синтаксического и лексического анализатора используются утилиты с открытым исходным кодом от проекта по разработке свободного программного обеспечения GNU Flex и Bison, написанных на языке C. Для создания веб-службы используется язык C# и кросс-платформенный веб-фреймворк Asp.Net Core. Целевой операционной системой являются ОС на ядре Linux (хотя возможно и выполнение на Windows и MacOS).

Главными задачами этой работы является создание веб-службы по синтаксическому разбору программ, написанных на языке C (по стандарту ANSI C99), которая будет иметь графический и программный интерфейс взаимодействия, а также создание синтаксического анализатора нацеленного на автоматическую генерацию блок-схем. Исходя из этих целей была выделена следующая необходимая функциональность сервисов:

а) аутентификация и авторизация с помощью графического интерфейса;
б) графический интерфейс взаимодействия с REST службой;
в) пользовательские роли с ограничениями пользования;
г) REST служба, включающая в себя:
    1) авторизация по ключу;
    2) интерфейс взаимодействия с парсером;
    3) поиск по проведенным операциям;
д) синтаксический разбор программ.

Целью данного курсового проекта является разработка веб-службы «ctox».

1 АНАЛИЗ ПРОТОТИПОВ, ЛИТЕРАТУРНЫХ ИСТОЧНИКОВ И ФОРМИРОВАНИЕ ТРЕБОВАНИЙ К ПРОЕКТИРУМОМУ ПРОГРАММНОМУ СРЕДСТВУ

1.1 Анализ существующих прототипов

Так как программное средство представляет собой комплекс взаимодействующих друг с другом модулей, для формирования требований к работе всей системы необходимо по отдельности рассмотреть различные прототипы существующих ПС с точки зрения:

– графического интерфейса;
– REST Api;
– парсинг сервиса.

1.1.1 Dribbble

Сайт Dribbble является популярной площадкой для артистов разных стилей: от художников до цифровых дизайнеров. В связи с большой посещаемостью сайта и специфичной тематикой его внешний вид разрабатывался, с целью эргономичности и удобства пользования.

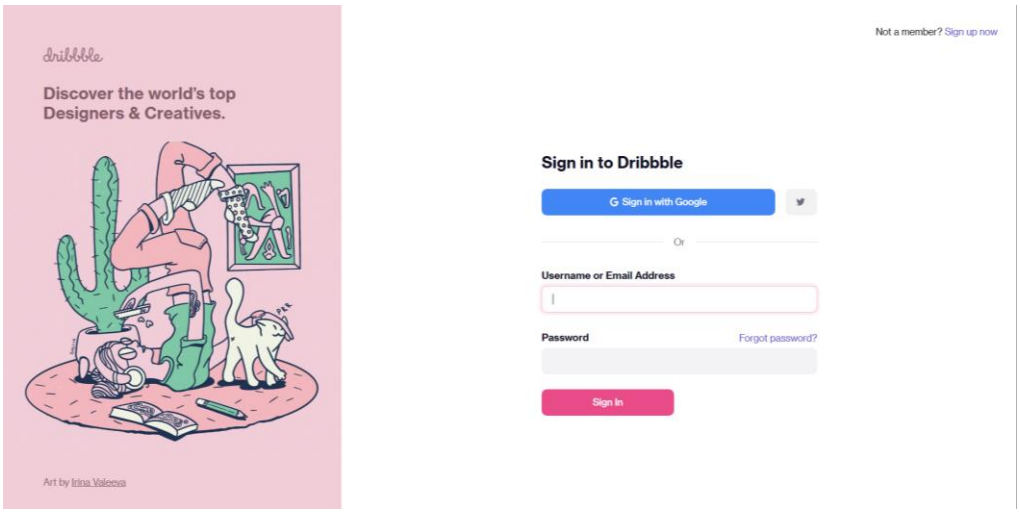

Рисунок 1.1 – Страница логина Dribble

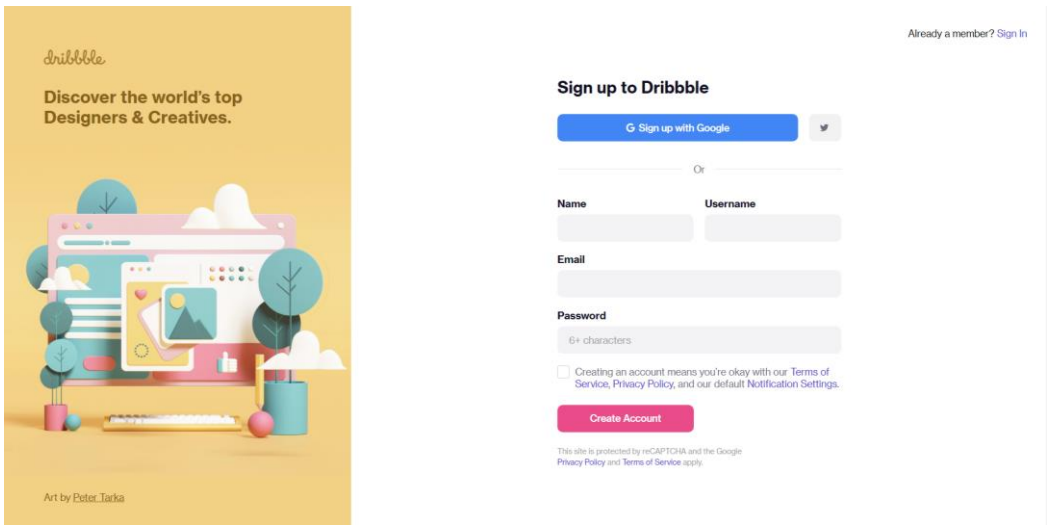

Рисунок 1.2 – Страница регистрации Dribbble

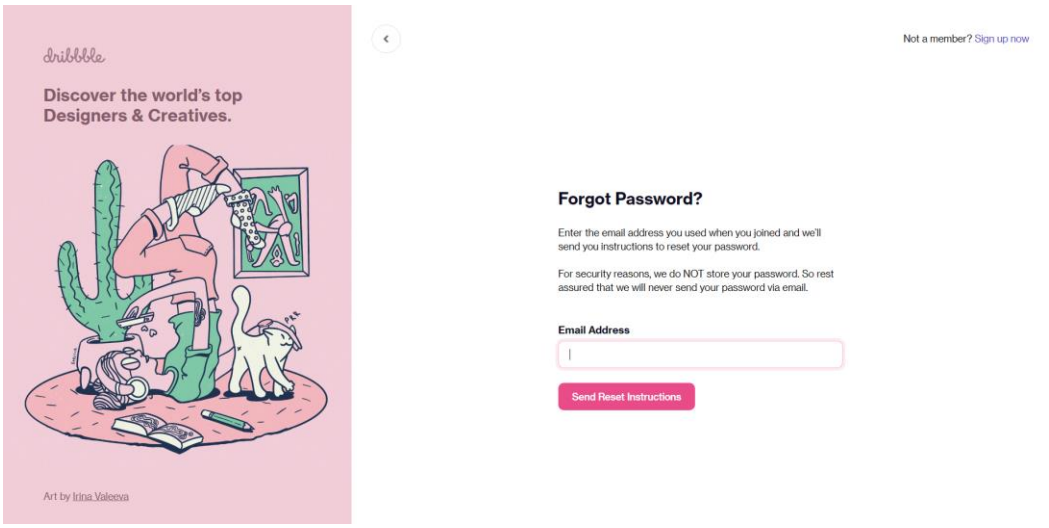

Рисунок 1.3 – Страница восстановления пароля Dribbble

Как видно из рисунков 1.1, 1.2 и 1.3 дизайн страниц аутентификации и авторизации пользователя выполнены в лаконичной и приятной цветовой гамме, все надписи понятны и полностью описывают ожидаемое поведение.

Из плюсов данного сайта можно отметить:

а) дружественный и красивый дизайн;
б) удобная форма регистрации и авторизации;
в) отсутствие потенциально уязвимых сообщений (например, при несовпадении аутентификационных данных не пишется, что конкретно введено неверно или при восстановлении пароля не выдается сообщение, о возможном отсутствии почты).

К минусам же можно отнести:

а) отсутствие удобного способа возврата на домашнюю страницу;
б) отсутствие поля повторного ввода пароля (при восстановлении пароля).

1.1.2 Google Cloud

Веб-служба от Google, предоставляющая сервисы SaaS, PaaS, IaaS, а также многие другие, имеет графический интерфейс для доступа к ним, что облегчает первоначальную установку и/или дает возможность пользования сервисами без прямого доступа к программному интерфейсу.

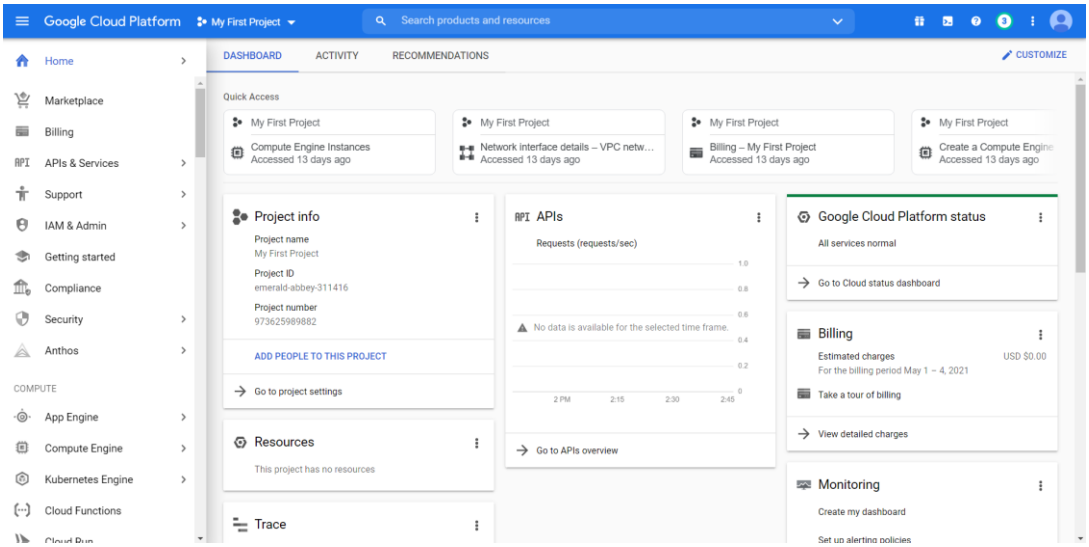

Рисунок 1.4 – Интерфейс Google Cloud

Данный сайт был выбран по причине того, что имеет схожую структуру: при существовании отдельного сервиса этот сайт также предоставляет графический доступ, что соответствует установленной функциональности.

К плюсам Google Cloud Console можно отнести:

а) удобный доступ к различным сервисам через меню;

б) единообразие интерфейса для различных видов аккаунтов.

Однако следует отметить, что данный интерфейс очень перегружен информацией, при выборе одного пункта меню открывается еще несколько, главная страница содержит большое количество нерелевантной информации.

1.1.3 Github Rest Api

Одним из наиболее примечательных примеров REST интерфейса на сегодняшний день является Github.

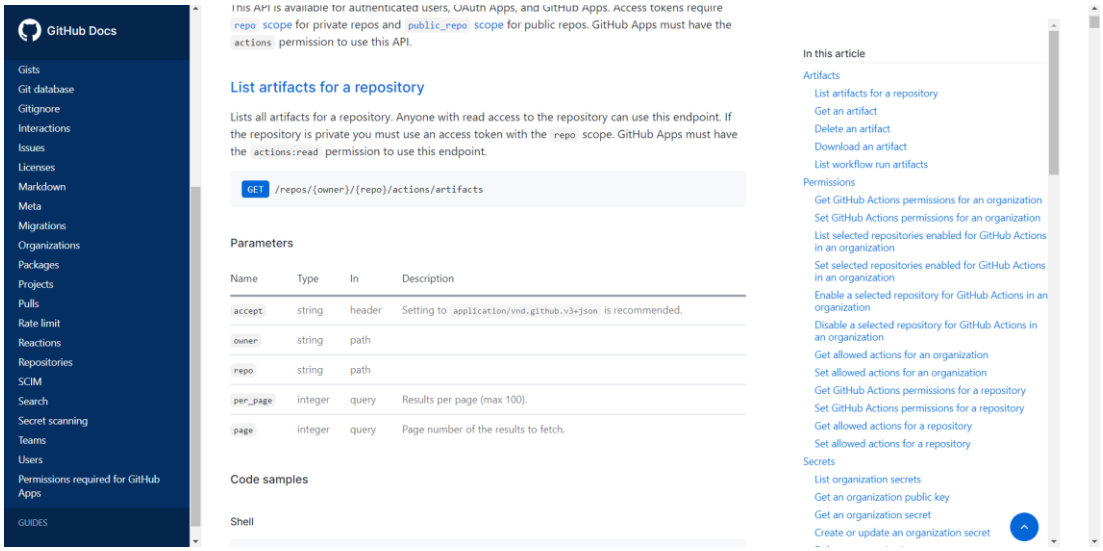

Рисунок 1.5 – Документация Github Rest

Github предоставляет огромное количество различных действий связанных с функциональностью этого сайта, что позволяет полностью управлять своим аккаунтом без графического интерфейса. Необходимо отметить, что для доступа к некоторым функциям необходима аутентификация через ключ. Являясь эталоном среди остальных поставщиков REST Api, можно отметить основные качества:

а) простая аутентификация посредством ключа и одного заголовка;
б) полная документация с описанием всех точек, параметров и ошибок;
в) иерархически логичная структура точек;
г) гибкая настройка результата.

1.1.4 Формальная грамматика языка C

Самой сложной частью построения парсера является формальная грамматика. Так как для создания синтаксического анализатора использует генератор парсеров (утилита, позволяющая из формальной грамматики получить готовый парсер), необходимо иметь грамматику для языка C стандарта ANSI C99 (данный стандарт был выбран по причине того, что в нем присутствуют все наиболее используемые синтаксические особенности языка, при этом отсутствуют более сложные, не поддающиеся синтаксическому разбору структуры), было решено использовать черновую версию 1995 года, находящуюся в открытом доступе. Данная версия покрывает стандарт практически на 100%, однако грамматика лишь содержит в себе основной каркас парсера и, как отмечает сам автор: «необходимо проделать еще много работы». Данная грамматика при использовании генератора создает синтаксическое дерево лишь условно, оно хранится в виде стека вызовов в

оперативной памяти, для дальнейшего использования необходимо расширить ее функциональность.

1.2 Постановка задачи

Основной задачей этой курсовой работы является создание веб-сервиса по синтаксическому разбору программ, написанных на языке C. Веб-сервис должен предоставлять как графический, так и программный интерфейс. Исходя из проанализированных прототипов, данное ПС должно иметь:

- Дружественный к пользователю графический интерфейс, оформленный в одном стиле для каждой страницы. Каждый элемент должен максимально полностью описывать свое поведение (например при помощи анимации или дополнительных подписей), а все сообщения об ошибках или дополнительной информации должны четко излагать основную суть, но при этом не разоблачать никаких уязвимых данных.
- Доступ к REST Api, через HTTP методы и авторизацию посредством выданного ключа. Каждый метод интерфейса должен излагать свое поведение в его названии и не быть подвержен ошибкам (т.е. должен проверяться любой пользовательский ввод). Возвращаемый результат зависит от авторизационного ключа, при этом на любое действие должен быть установлен, семантически верный, код результата.
- Сервис синтаксического разбора, имеющий возможность взаимодействовать с остальной программой и выводящий результат в виде дерева XML, при этом должен самостоятельно обрабатывать лексические и синтаксические ошибки.

2 АНАЛИЗ ТРЕБОВАНИЙ К ПРОГРАММНОМУ СРЕДСТВУ И РАЗРАБОТКА ФУНКЦИОНАЛЬНЫХ ТРЕБОВАНИЙ

2.1 Описание функциональности программного средства

Данное ПС запускается как сервер, прослушивающий определенный порт, взаимодействующий с сервисами по протоколу HTTP (при желании этот порт можно скрыть из публичного доступа и настроить реверс-прокси для работы по защищенному соединению). Веб-сайт имеет домашнюю страницу с основной информацией о веб-службе. Язык интерфейса – английский.

2.1.1 Графический интерфейс

Веб-сайт предоставляет доступ пользователю к регистрации, авторизации и восстановлению пароля. После успешной авторизации отображается основное навигационное меню, предоставляющая возможность к пользованию программным интерфейсом (возможно с неполным функционалом). Все запросы, не являющиеся необходимыми для отображения страниц, происходят асинхронно, при помощи использования JS библиотек (например JQuery), при этом страница не «замирает».

2.1.2 Программный интерфейс

Доступ к REST Api происходит по протоколу HTTP, через отдельно выделенный путь «/api». Авторизация происходит по ключу в заголовке расширения «x-api-key». Существует тестовая точка для проверки соединения и работоспособности сервиса. Параметры в запрос передаются: в строке запроса – если они не большие и не содержат уязвимой информации; в теле запроса, в JSON формате – если они объемные и/или содержат уязвимую информацию (таким образом можно обеспечить защищенность, при использовании реверс-прокси). Каждый запрос на парсинг сохраняется, для последующего просмотра в истории.

2.1.3 Сервис синтаксического анализа

Основное ПС взаимодействует с заранее скомпилированным парсером через стандартные потоки ввода-вывода, чтобы обеспечить максимальную скорость и уменьшить количество возможных ошибок внедрения разных систем. При этом анализатор выводит ошибки также в стандартный поток вывода, с тем же синтаксисом, что и используется для вывода абстрактного синтаксического дерева.

2.2 Спецификация функциональных требований

Во время разработки данного программного средства должны быть реализованы следующие функции:

а) графический интерфейс:
   1) регистрация с подтверждением почты;
   2) авторизация;
   3) восстановление пароля;
   4) синтаксический анализ (доступ к программному интерфейсу);
   5) просмотр истории;
   6) документация к REST Api;

б) программный интерфейс:
   1) тестовая точка доступа;
   2) авторизация по ключу;
   3) парсинг;
   4) просмотр истории запросов;

в) сервис синтаксического разбора с построением абстрактного синтаксического дерева в виде дерева XML.

3 ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА

3.1 Модуль авторизации и аутентификации

Самой основной функциональностью веб-сайта (графической частью сервиса) является модуль авторизации и аутентификации, ведь только через него можно получить доступ к программному интерфейсу (а т.е. к главной функциональности сайта). Помимо этого он также помогает избежать большой нагруженности веб-сервиса, путем установки ограничения на частоту использования (с помощью авторизации).

3.1.1 Авторизация на основе Claim

Существует два основных вида авторизации пользователя – ролевая и на основе claim. При использовании первого типа учетные данные пользователя, необходимые для аутентификации, передаются при каждом запросе к серверу. Помимо того, что это замедляет работу сервера, т.к. при каждом запросе необходимо обращаться к базе данных, это также расширяет возможности для атак, увеличивая промежуток времени, в котором можно перехватить уязвимые данные. По этой причине в данном курсовом проекте был выбран второй тип авторизации – на основе Claim.

Claim – некоторый кусок информации, содержащий в себе авторизационные данные (например роль, дату рождения, локацию и т.д.). Таким образом аутентифировать пользователя необходимо только один раз (при самом первом запросе), далее ему выдается куки, с сериализованными и подписанными claim, по которым, в следствии, происходит авторизация. В качестве авторизационных данных во всем приложении достаточно лишь двух claim – имени пользователя и роли. Ниже представлен код, отвечающий за аутентификацию.

```
// Асинхронный метод аутентификации
private Task Authenticate(User user)
{
    // Установка claim - имя пользователя и роли
    var claims = new List<Claim>
    {
        new Claim(ClaimsIdentity.DefaultNameClaimType, user.Username),
        new Claim(ClaimsIdentity.DefaultRoleClaimType, user.Role.ToString()),
    };

    var id = new ClaimsIdentity(claims, "ApplicationCookie",
        ClaimsIdentity.DefaultNameClaimType,
        ClaimsIdentity.DefaultRoleClaimType);

    // Установка куки со всеми claim
    return HttpContext.SignInAsync(
                CookieAuthenticationDefaults.AuthenticationScheme,
                new ClaimsPrincipal(id));
}
```

Используя данную имплементацию от Microsoft (предоставленная вместе с фреймворком Asp.Net Core), всю последующую аутентификацию можно производить одним аттрибутом «Authorize». Для проверки авторизационных данных необходимо установить свойство «Policy» (по умолчанию существует свойство «Roles», которое проверяет claim Role).

3.1.2 Структура авторизации и аутентификации

Алгоритм регистрации с подтверждением почты, аутентификации и восстановления пароля, а также структура хранения всех авторизационных данных является тривиальной задачей, со множеством решений и советов – что можно и нельзя делать. В рамках этого курсового проекта было решено не уходить далеко от всеми проверенными алгоритмами:

- таблица с пользователями содержит поля «имя», «почта», «хэш пароля» и «подтвержден» (помимо других необходимых);
- пароль дважды хэшируется, используя алгоритм SHA256, добавляя соль в виде имени пользователя;
- таблица «подтверждения» содержит ссылку на пользователя и случайную строку;
- при регистрации случайная строка отправляется вместе с ссылкой на подтверждение на почту, при совпадении строк пол «подтвержден» устанавливается в значение true;
- таблица «восстановления» содержит ссылку на пользователя, случайную строку и поле «действительно»;
- при восстановлении пароля случайная строка вместе с ссылкой на восстановление отправляется на почту, при совпадении строк пароль заменяется, а поле «действительно» устанавливается в значение false.

Ниже приведена более подробная блок-схема этих алгоритмов, за исключением деталей имплементации (непосредственно связанных с фреймворком и отправкой письма). Алгоритмы подтверждения и восстановления приведены в приложении В.

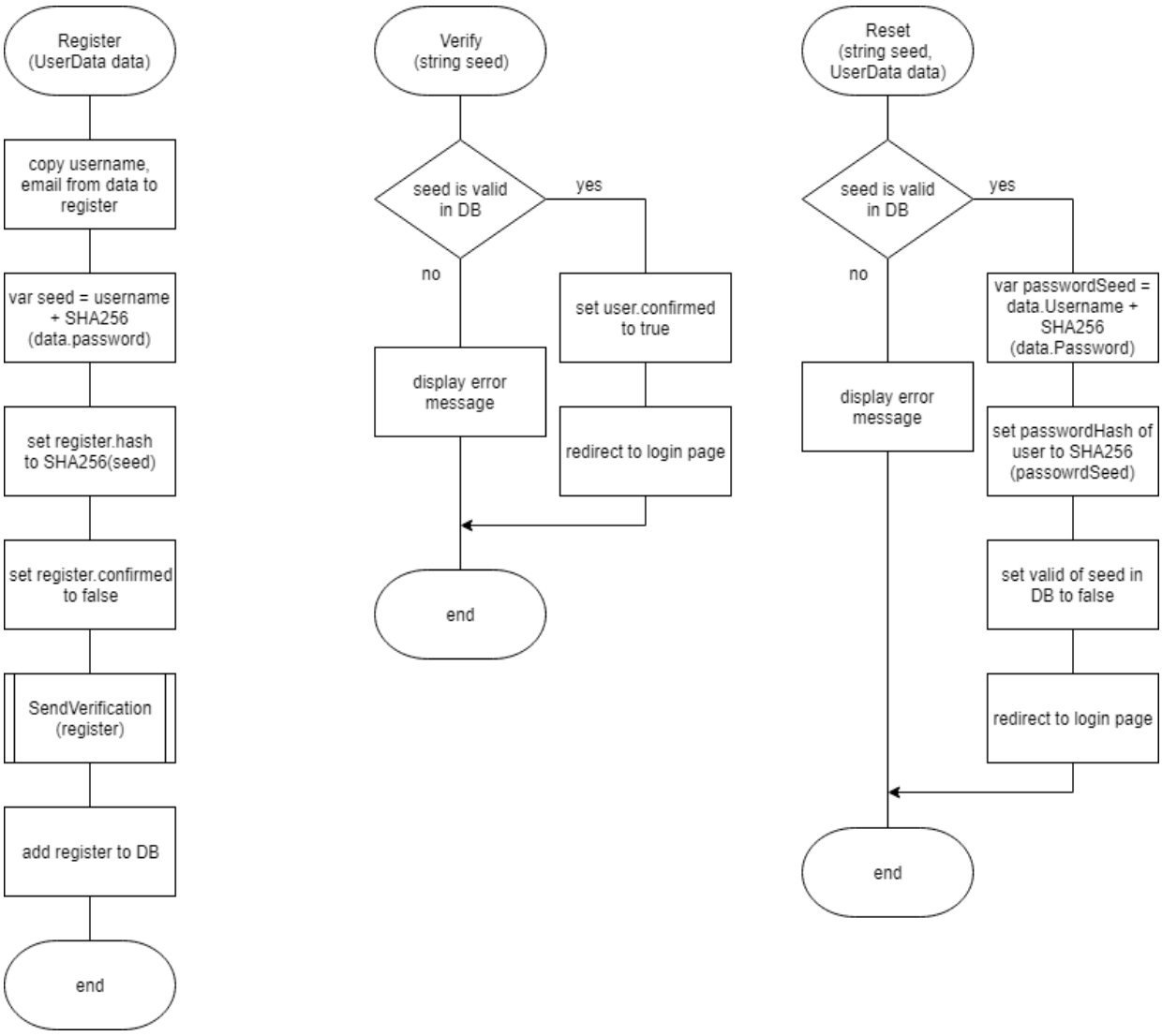

Рисунок 3.1 – Алгоритмы аутентификации и авторизации

3.2 Модуль REST Api

Данная часть является сердцем ПС, так как оно содержит самую основную бизнес логику, через которое взаимодействуют практически все сервисы курсового проекта. Помимо того, что это модуль предоставляет конечные точки, на которые пользователь должен обращаться, он также оперирует с базой данных, сервисом ограничений, сервисом парсинга, сервисом авторизации по ключу и сервисом сжатия данных (по сути дела имплементируя микросервисную архитектуру).

Сервис ограничений выполняет две проверки – последнее обращение к сервису парсинга и размер исходного файла. Эти ограничения задаются через файл конфигурации для каждого типа роли отдельно. При каждом обращении к сервису устанавливается время, после чего оно сравнивается – если прошло пройденное время больше, чем установленное ограничение, управление передается модулю, в ином случае возвращается результат с кодом 429 Too

Many Requests. Такая же проверка существует и на размер исходного файла в байтах – код результата, при его превышении – 413 Payload Too Large.

Сервис сжатия вызывается при каждом запросе – исходный и результирующий файлы сжимаются с помощью алгоритма GZip, а затем раскрываются при обращении к истории. Так как большая часть останльных методов – тривиальна, вся логика находится в сервисах, то в этом разделе отдельно следует рассмотреть сервис авторизации.

3.2.1 Сервис авторизации

Как уже было сказано выше – авторизация проходит по случайно-генерируемому ключу, который в следствии передается в заголовке расширения «x-api-key». Сервис авторизации выполнен в виде фильтра действий – он срабатывает **до** передачи управления в модуль и проверяет наличие и корректность ключа. При валидности данных фильтр пропускает запрос, при невалидности – останавливает обработку и возвращает код результата 401 «Unauthorized». Для более удобного использования этот сервис также имплементирует интерфейс атрибута (чтобы его можно было легко подключать) и проверяет наличие атрибута «AllowNoApi» при валидации (в целях отмены действия фильтра, например при авторизации с использованием claim). Ниже представлена вся логика фильтра.

```
public void OnActionExecuting(ActionExecutingContext context)
{
    // Если установлен атрибут AllowNoApi, ничего не делать
    if ((context.ActionDescriptor as ControllerActionDescriptor)
        .MethodInfo
        .GetCustomAttributes(typeof(AllowNoApi), false)
        .FirstOrDefault() != null) return;

    // Если нет заголовка - вернуть 401
    if (!context.HttpContext.Request.Headers.ContainsKey("x-api-key"))
    {
        context.Result = new UnauthorizedResult();
        return;
    }

    var dbContext = (AppDbContext)Context.HttpContext
        .RequestServices.GetService(typeof(AppDbContext));

    var api = dbContext.Apis.Include(a => a.User).FirstOrDefault
        (a => a.Key.Equals
            (context.HttpContext.Request.Headers["x-api-key"]));

    // Если ключ неверен - вернуть 401
    if (api is null)
    {
       context.Result = new UnauthorizedResult();
       return;
    }
}
```

3.3 Модуль парсинга

Как уже было отмечено ранее, парсинг исходных блоков программ производится заранее скомпилированным парсером, с которым ПС взаимодействует через стандартные потоки ввода/вывода (утилита считывает данные с потока ввода до тех пор, пока поток открыт). Данный подход был выбран по причине гибкости и расширяемости. В первую очередь это связано с выбором утилит-генераторов лексера и парсера. Так как существует большой выбор генераторов, то, при их замене и использовании компилятора под .Net фреймворк, необходимо было бы переписать большое количество кода, связывающего ПС и модуль. Во-вторых, имея отдельно скомпилированный модуль, его можно независимо от основной программы заменять и/или модифицировать, без необходимости перекомпилирования неизменных частей. Таким образом, при расширении сервиса до, к примеру, построения блок-схем, количество кода, которое необходимого изменить, будет минимальным. Однако такой подход негативно сказывается на общей скорости приложения, т.к. запускать отдельно второй процесс, при этом обмениваясь данными через стандартные потки, может быть относительно медленным, однако выигрыш от скорости написания ПС и удобства его содержания гораздо превышают негативные эффекты. С учетом всего вышесказанного, связать парсер и ПС можно с помощью класса Process.

```
public string Parse(string content)
{
    // Создание процесса, с перенаправленными стандартными потоками
    using var process = new Process();
    process.StartInfo.FileName = command;
    process.StartInfo.RedirectStandardInput = true;
    process.StartInfo.RedirectStandardOutput = true;

    // Запуск процесса и запись в поток ввода
    process.Start();
    process.StandardInput.Write(content);

    // Закрытие потока ввода
    process.StandardInput.Close();

    // Закрытие процесса и сохранение вывода
    var result = process.StandardOutput.ReadToEnd();
    process.WaitForExit();

    return result;
}
```

4 СОЗДАНИЕ ПРОГРАММНОГО СРЕДСТВА

В ходе разработки программного средства курсовой проект был разделен на 2 подпроекта: сервис парсинга и веб. Сервис парсинга включает в себя:

а) Ctox – класс-обертка для работы с парсером:
   1) Parse – принимает в качестве ввода исходный код программы, возвращает синтаксическое дерево;
   2) ParseAsync – асинхронный вызов метода Parse;
б) c_parser.tab.h – заголовочный файл парсера с лексемами (в виде перечисляемого типа);
в) c_parser.tab.c, c_parser.lex.c – исходный код парсера и лексера соответственно.

Веб служба выполнена в соответствии с паттерном Model-View-Controller (паттерн взаимодействия с пользователем, позволяющий разделить логику работы и представление данных), тем самым разделив все классы проекта на следующие категории:

а) attributes – помощники, облегчающие написание моделей и конроллеров;
б) controllers – взаимодействие с пользовательским вводом;
в) dal – работа с данными;
г) extensions – расширения типов;
д) tag helpers – помощники, облегчающие написание представлений;
е) models – структуры данных работы с пользователем;
ж) services – сервисы с основной логикой;
з) views – представления;
и) wwwroot – статические файлы представления.

Для каждой категории следует рассмотреть интерфейсы классов:

а) атрибут ApiKey – фильтр запросов по Api ключу:
   1) свойство Roles – допустимые роли;
   2) метод OnActionExecuting – проверка доступа **до** передачи управления контроллеру;
б) атрибут AllowNoApi – класс-маркер, отменяющий действия ApiKey;
в) атрибут EmailAttribute – класс-валидатор почты, с методом IsValid;
г) атрибут StringPasswordAttribute – класс-валидатор пароля, с установленными ограничениями (наличие специальных символов, чисел и т.д.), с методом IsValid;
д) атрибуты UniqueEmailAttribute и UniqueUsernameAttribute – класс-валидатор уникальности имена с методом IsValid;
е) контроллер AdminApiController – контроллер Api запросов с ключом администратора, со следующими точками:
   1) GetUsers – получение списка всех пользователей;

2) DeleteUser – удаление пользователя по идентификатору;
3) GetStat – получение дополнительной информации о пользователе (количество восстановлений, запросов и т.д.);
4) SetConfirmed – подтверждение пользователя без почты;
5) SetRole – установка роли;
6) GetConversions – получение всех переводов пользователя;
7) DeleteConversion – удаление конкретного перевода;
8) ViewConversion – просмотр конкретного перевода;

ж) контроллер ApiController – контроллер Api запросов для пользователей, с методами:
1) Test – тестовая точка для проверки соединения;
2) Create – создание и/или пересоздание ключа api (доступно только из графического интерфейса);
3) Parse – парсинг;
4) History – просмотр истории с возможностью настройки поиска;
5) View – просмотр конкретного перевода;

з) контроллер AuthController – контроллер управления авторизации и аутентификации пользователя, со следующими методами:
1) Login – вывод представления со страницей авторизации или аутентификация пользователя;
2) Register – вывод представления регистрации или регистрация пользователя;
3) Verify – подтверждение почты пользователя по коду;
4) Restore – вывод страницы восстановления или отправка кода восстановления пользователю;
5) Reset – смена пароля;
6) Logout – деаутентификация;

и) контроллер ErrorController – контроллер обработки ошибок

к) контроллер HomeController – контроллер управления основного меню взаимодействия, с функциями:
1) Index – вывод представления с конвертацией (парсингом) или отправка запроса на Api;
2) History – вывод представления с ограниченной историей (только 10 последних операций);
3) Subscription – вывод представления с информацией о типах аккаунтов (время задержки, максимальный размер и т.д.);
4) Unsubscribe – понижение типа аккаунта то минимального;
5) Upgrade – повышение типа аккаунта до «платного»;
6) Api – вывод представления с документацией об api;

л) AppDbContext – контекст базы-данных;

м) ClaimsIdentitityExtensions – класс расширения класса ClaimsIdentity и метод UpdateClaim – замена значения одного claim на другой;

н) RouterTagHelper – класс-помощник представлений, отрисовывающий «активную» ссылку в навигационном меню с определенным css классом, по названию страницы;
о) модели уровня домена – модели данных, образующих таблицы в БД:
    1) Api – таблица с информацией о ключе, последнем использовании и ссылкой на пользователя;
    2) Conversion – таблица с датой перевода, ее содержании, типе и ссылкой на пользователя;
    3) User – таблица с именем пользователя, его почтой, паролем и ролью;
    4) PasswordRestore – таблица с ключом восстановления пароля и ссылкой на пользователя;
    5) UserVerification – таблица с ключом подтверждения почты пользователя и ссылкой;
п) модели уровня представления – модели данных, структурирующие данные передаваемые от пользователя контроллерам:
    1) ParseRequest – тип перевода и исходный код;
    2) Conversion – данные для заполнения таблицы;
    3) ErrorInfo – код и сообщение ошибки;
    4) UserLogin – имя пользователя и пароль;
    5) UserRegister – имя пользователя, пароль и почта;
    6) UserRestore – пароль и подтверждение;
р) сервис EmailSenderService – сервис отправки почты с аккаунта веб-службы (метод SendEmail);
с) сервис GzipCompressService – сервис сжатия и декомпрессии текстовых данных (методы Compress и Decompress);
т) сервис HashService – сервис хэширования и генерации случайных строк (Hash, HashHttpSafe, RandomHash);
у) сервис ParseService – сервис-адаптер для сервиса парсинга;
ф) сервис RestrictionService – сервис проверки условий пользования веб-службой (IsAllowedTimeout – проверка задержки между запросами на парсинг; IsAllowedSize – проверка максимального размера файла);
х) представления:
    1) Login – страница входа;
    2) Register – страница регистрации;
    3) Reset – страница восстановления пароля;
    4) Restore – страница смена пароля;
    5) Error – страница с информацией об ошибке;
    6) Api – страница с документацией об api;
    7) Convert – страница с переводом кода;
    8) History – страница с таблицей истории;
    9) Subscription – страница с выбором типа аккаунта;
ц) default.html – лэндинг сайта.

5 ТЕСТИРОВАНИЕ, ПРОВЕРКА РАБОТОСПОСОБНОСТИ И АНАЛИЗ ПОЛУЧЕННЫХ РЕЗУЛЬТАТОВ

5.1 Тестирование, проверка работоспособности и анализ полученных результатов

Тестирование программного средства производилось на персональном компьютере с установленной операционной системой Windows 10 и браузером Google Chrome версии 90.0.

Таблица 5.1 – Тестовые сценарии авторизации аутентификации

№	Тестируемая функциональность	Последовательность действий	Ожидаемый результат	Полученный результат
1	2	3	4	5
1	Регистрация по электронной почте	1) Открыть главную страницу 2) Выбрать пункт «Login» 3) Выбрать пункт «Register now» 4) Ввести необходимые данные 5) Перейти на указанную почту 6) Открыть письмо от сервиса 7) Перейти по отправленной ссылке	Отображение сообщения об успешном подтверждении аккаунта	Тест пройден
2	Регистрация с не валидным именем пользователя	1) Открыть главную страницу 2) Выбрать пункт «Login» 3) Выбрать пункт «Register now» 4) Ввести не валидное имя пользователя (уже существующее или содержащие не алфавитные символы)	Отображение сообщения с указанием ошибки	Тест пройден

Продолжение таблицы 5.1

1	2	3	4	5
3	Регистрация с не валид ной почтой	1) Открыть главную страницу 2) Выбрать пункт «Login» 3) Выбрать пункт «Register now» 4) Ввести не валидную почту (уже существующую или не корректную)	Отображение сообщения с указанием ошибки	Не валидная почта принимается сервером и, при попытке отправить письмо, выдает исключение. **Тест не пройден**
4	Регистрация с не валид ным паролем	1) Открыть главную страницу 2) Выбрать пункт «Login» 3) Выбрать пункт «Register now» 4) Ввести не валидный пароль	Отображение сообщения с указанием ошибки	Тест пройден
5	Авторизация по имени пользователя	1) Открыть главную страницу 2) Выбрать пункт «Login» 3) Ввести имя пользователя и пароль 4) Нажать кнопку «Sign in»	Отображение страницы с указанием текущего пользователя	Тест пройден
6	Авторизация с не валид ным именем пользователя или паролем	1) Открыть главную страницу 2) Выбрать пункт «Login» 3) Ввести не валидное имя пользователя или пароль	Отображение сообщении о не существую щем аккаунте	Тест пройден
7	Восстановле ние пароля по электрон ной почте	1) Открыть главную странциу 2) Выбрать пункт «Login» 3) Выбрать пункт «Forgot password»	Отображение сообщения об успешной смене пароля	Тест пройден

Продолжение таблицы 5.1

1	2	3	4	5
		4) Ввести почту ассоциируемую с аккаунтом 5) Перейти на указанную почту 6) Открыть письмо от сервиса 7) Перейти по ссылке 8) Указать новый пароль 9) Нажать кнопку «Reset»		
8	Восстановление пароля с не существующей почтой	1) Открыть главную странциу 2) Выбрать пункт «Login» 3) Выбрать пункт «Forgot password» 4) Ввести не существующую почту	Отображение сообщения без ошибок	Тест пройден
9	Восстановление пароля с не правильным паролем	1) Открыть главную странциу 2) Выбрать пункт «Login» 3) Выбрать пункт «Forgot password» 4) Ввести почту ассоциируемую с аккаунтом 5) Перейти на указанную почту 6) Открыть письмо от сервиса 7) Перейти по ссылке 8) Указать неверный пароль	Отображение сообщение об ошибке	Тест пройден

Для проведения последующих тестов связанных с основной функциональностью сайта, необходимо произвести авторизацию по шагам, указанных в 5 пункте таблицы 5.1.

Таблица 5.2 – Тестовые сценарии основной функциональности

№	Тестируемая функциональность	Последовательность действий	Ожидаемый результат	Полученный результат
1	2	3	4	5
1	Парсинг кода	1) Выбрать пункт меню «Convert» 2) Ввести исходный код 3) Выбрать тип перевода 4) Нажать кнопку «Parse»	Отображение синтаксического дерева в выбранном виде	Тест пройден
2	Парсинг пустого кода	1) Выбрать пункт меню «Convert» 2) Ввести пустой код 3) Нажать кнопку «Parse»	Отображение ошибки о пустом коде	Тест пройден
3	Парсинг кода с интервалом меньшим, чем позволено аккаунту	1) Выбрать пункт меню «Convert» 2) Ввести код 3) Нажать кнопку «Parse» дважды	Отображение дерева при первом переводе, отображение ошибки при втором	Тест пройден
4	Парсинг кода размера большим, чем позволено аккаунту	1) Выбрать пункт меню «Convert» 2) Ввести код больше, чем 1Кбайт 3) Нажать кнопку «Parse»	Отображение ошибки о большом размере кода	Тест пройден
5	Парсинг кода с лексической ошибкой	1) Выбрать пункт меню «Convert» 2) Ввести символ, не из алфавита исходного кода 3) Нажать кнопку «Parse»	Отображение синтаксического дерева с информацией об ошибке	Сервис игнорирует лексические ошибки. **Тест не пройден**
6	Парсинг кода с синтаксической ошибкой	1) Выбрать пункт меню «Convert» 2) Ввести код с синтаксической ошибкой 3) Нажать кнопку «Parse»	Отображение синтаксического дерева с информацией об ошибке	Тест пройден

Продолжение таблицы 5.2

1	2	3	4	5
7	Парсинг кода без ключа Api	1) Выбрать пункт меню «Convert» 2) Ввести код 3) Нажать кнопку «Parse»	Отображение ошибки об отсутствии ключа	Тест пройден
8	Просмотр истории	1) Выбрать пункт меню «History»	Отображение до 10 последних запросов	Отображение **первых** 10 запросов **Тест не пройден**
9	Просмотр конкретного перевода	1) Выбрать пункт меню «History» 2) Выбрать один из переводов и нажать кнопку «View»	Отображение исходного кода и результата с правильным типом	Тест пройден
10	Просмотр подписки	1) Выбрать пункт меню «Subscription»	Отображение страницы с выбором подписки или ее отмены	Тест пройден
11	Выбор подписки «Pro»	1) Выбрать пункт меню «Subscription» 2) Нажать кнопку «Upgrade»	Отображение нового типа аккаунта	Тип аккаунта не меняется **Тест не пройден**
12	Отмена подписки	1) Выбрать пункт меню «Subscription» 2) Нажать кнопку «Unsubscribe»		
13	Генерация ключа api	1) Выбрать пункт меню «Api» 2) Нажать кнопку «Generate»	Отображение новое ключа api	Тест пройден
14	Копирование ключа api	1) Выбрать пункт меню «Api» 2) Нажать кнопку копирования	Копирование ключа в буфер обмена	Тест пройден

Как видно из таблицы 5.2 – не все тесты были пройдены успешно, однако все они тривиальны:

– Стандартная имплементация проверки почты на валидность от Asp.Net Core заключалась в единственной проверке на присутствие символа «@». Замена на более точное регулярное выражение позволило отбросить большее количество не валидных данных.
– Лексер при встрече символов, не из алфавита языка, пропускал их и переходил к следущему. Решением является добавление вызова функций обработки ошибок, с наименьшим коэффициентом (в самом конце файла).
– История при запросе в БД лимитировала количество вывода до 10, однако по умолчанию записи отсортированы по ключу (а т.е. и по дате создания). Сменив вид сортировки на по убыванию, страница начала выдавать правильный ожидаемый результат.
– При смене подписки куки, хранящие в себе данные о типе аккаунта, не обновлялись вместе с самим аккаунтом. Отправка заголовка на установку нового значения куки, решило обе проблемы сразу.

После прохождения тестирования и устранения выявленных ошибок, основные возможности ПС стали доступными для пользователя, а их функциональность подтверждена, как работающая.

6 РУКОВОДСТВО ПО УСТАНОВКЕ И ИСПОЛЬЗОВАНИЮ

6.1 Установка программного средства

Данное программное обеспечение оптимизировано для работы со средством контейнеризации Docker. Среда должна включать в себя:

– СУБД MySql;
– Реверс-прокси сервер.

Пример файла docker-compose можно увидеть ниже

```
version: '3'

services:
  mysql:
    image: mysql:8
    environment:
      - "MYSQL_DATABASE=ctox_db"
      - "MYSQL_ROOT_PASSWORD=12345"
    volumes:
      - mysqldata:/var/lib/mysql

  ctoxweb:
    image: madopew/ctoxwebapp:latest
    depends_on:
      - mysql
    environment:
      - "ConnectionStrings:AppContext=server=mysql; port=3306;
database=ctox_db; user=root; password=12345"
      - "Email:Name=email@gmail.com"
      - "Email:Password=strongPassword"
    links:
      - mysql
    volumes:
      - aspnetcorekeys:/root/.aspnet/DataProtection-Keys # data protection
keys shared volume
    restart: on-failure # in case mysql not loaded yet

  reverse-proxy:
    image: nginx:latest
    depends_on:
      - ctoxweb
    ports:
      - 80:80
    volumes:
      - ./proxy.conf:/etc/nginx/conf.d/default.conf

volumes:
  mysqldata:
  aspnetcorekeys:
```

6.2 Руководство пользователя

6.2.1 Регистрация

При первом использовании веб-сервиса пользователю необходимо произвести регистрацию и подтверждение электронной почты. Для этого следует произвести следующие действия:

1) На главной страница сайта нажать на кнопку «Login» в правом-верхнем углу экрана.
2) На появившейся странице нажать на кнопку «Register now».
3) Отобразиться следующая страница:

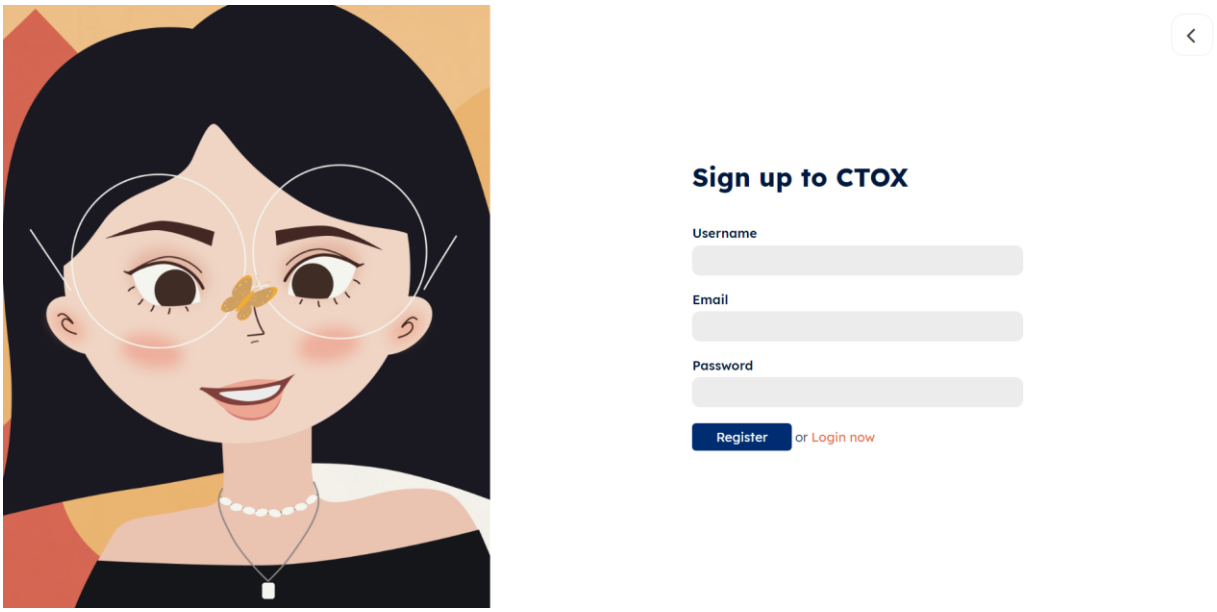

Рисунок 6.1 – Страница регистрации

4) В поля «Username», «Email», «Password» необходимо ввести: имя пользователя, почту и пароль, соответственно. На данном этапе следует обратить внимание на появляющиеся сообщения выделенным красным цветом – они содержат информацию о возможных невалидных данных.
5) Нажать кнопку «Register».

После выполнения этих действий, на указанную ранее почту придет письмо, с дальнейшими указаниями по подтверждению почты. Необходимо перейти по ссылке.

6.2.2 Авторизация

После подтверждения почты отобразится следующая страница:

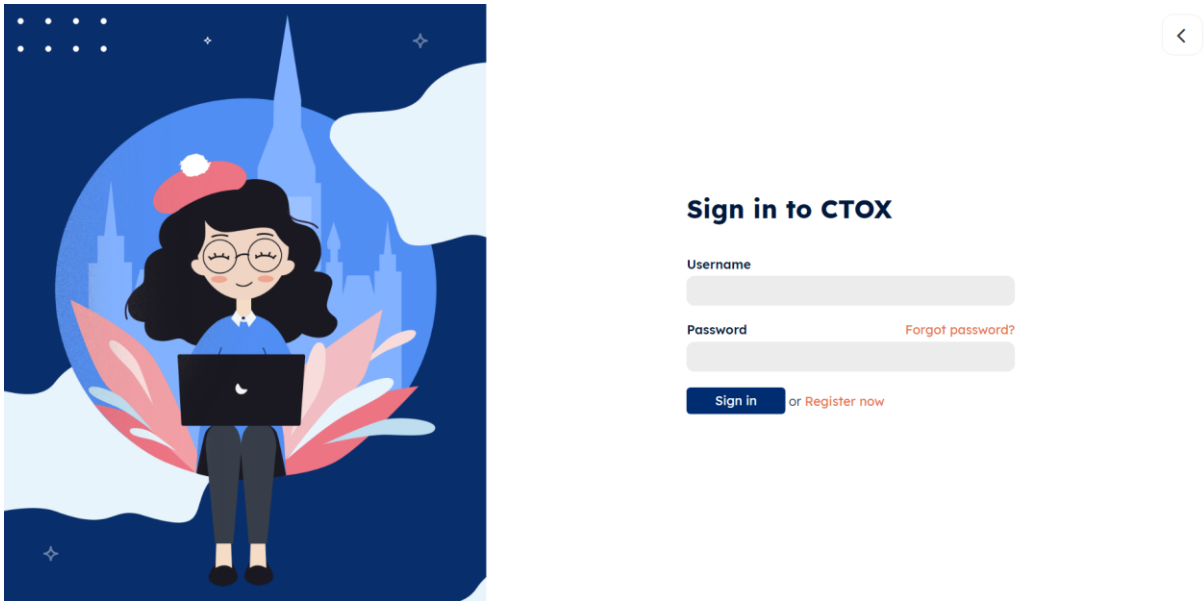

Рисунок 6.2 – Страница авторизации

Для проведения авторизации следует ввести данные, ранее введенные на странице регистрации.

6.2.3 Использование сервиса

Доступ к сервису можно осуществлять посредством графического и программного интерфейса. Графический интерфейс представляет собой веб-сайт с основной (однако ограниченной) функциональностью. Программный интерфейс (выполненный в виде REST Api) представляет более гибкую систему, с большим спектром настроек. Для использования обеих видов доступа необходимо произвести авторизацию.

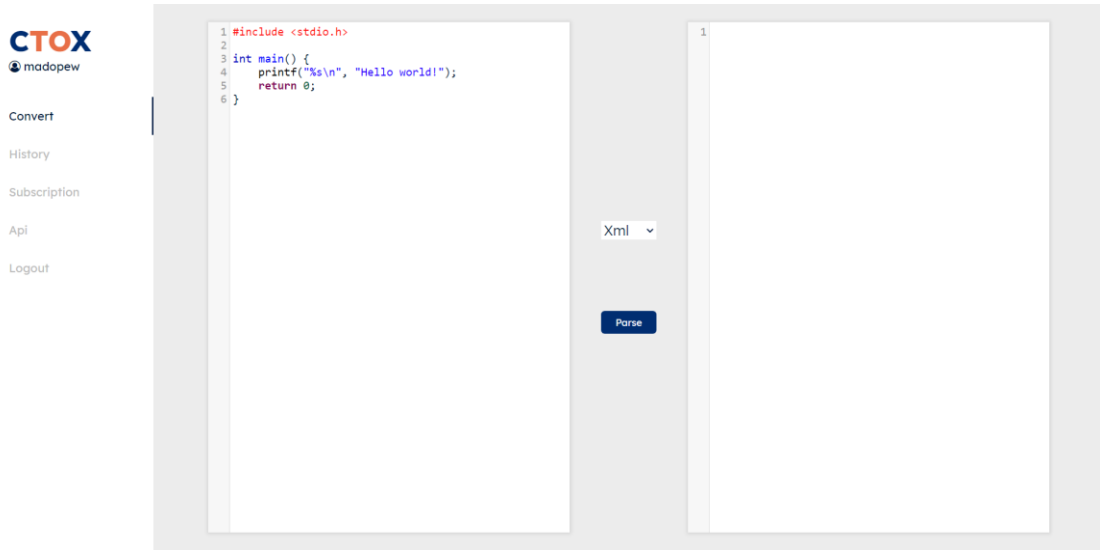

Рисунок 6.3 – Страница веб-сервиса

Графический интерфейс состоит из двух основных частей – меню и окна взаимодействия. В меню представлены следующие пункты:

1) Convert – парсинг программ, с веб-редактором кода;
2) History – просмотр последних запросов;
3) Subscription – информация о типе аккаунта;
4) Api – документация к REST Api;
5) Logout – деавторизация (выход из аккаунта).

Далее рассмотрены все пункты.

6.2.3.1 Парсинг

Для доступа к этой секции необходимо выбрать пункт меню «Convert».

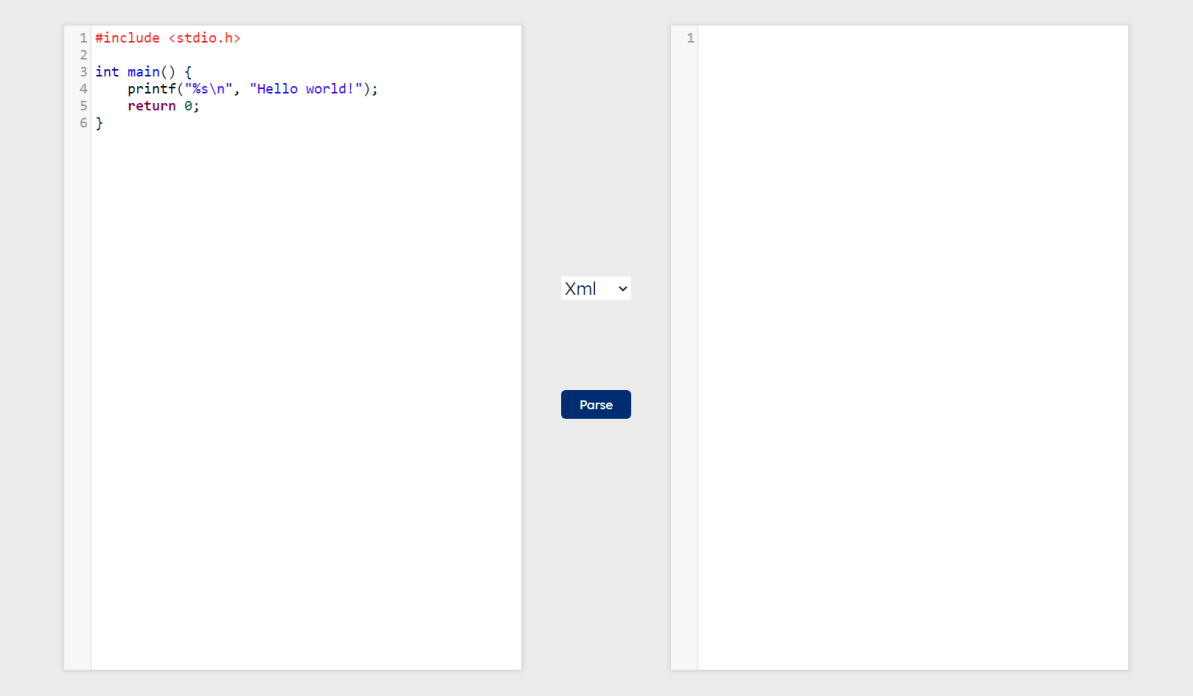

Рисунок 6.4 – Окно парсинга

В данной секции находятся:

1) Редактор кода – окно с подсветкой синтаксиса и автоматической индентацией, для более удобного ввода и изменения исходного кода;
2) Выпадающий список типа результата – формат выходного файла;
3) Кнопка;
4) Окно просмотра результата – окно с подсветкой синтаксиса и автоматической индентацией, для более удобного просмотра результата.

Для парсинга необходимо ввести исходные данные в редактор кода, выбрать формат выходного файла и нажать кнопку «Parse».

6.2.3.2 История

Для доступа к этой секции необходимо выбрать пункт меню «History».

Рисунок 6.5 – Окно истории

В данной секции отображаются до 10 последних операций. В основной части экрана отображается таблица со следующими полями:

1) Date – дата конвертации;
2) Time – время конвертации;
3) Parsing type – формат конвертации;
4) Display acion – кнопка просмотра полной информации.

Для того чтобы просмотреть определенный запрос, необходимо нажать на соответствующую кнопку «View».

Рисунок 6.6 – Информация о запросе

6.2.3.3 Подписка (тип аккаунта)

Для доступа к этой секции необходимо выбрать пункт меню «Subscription».

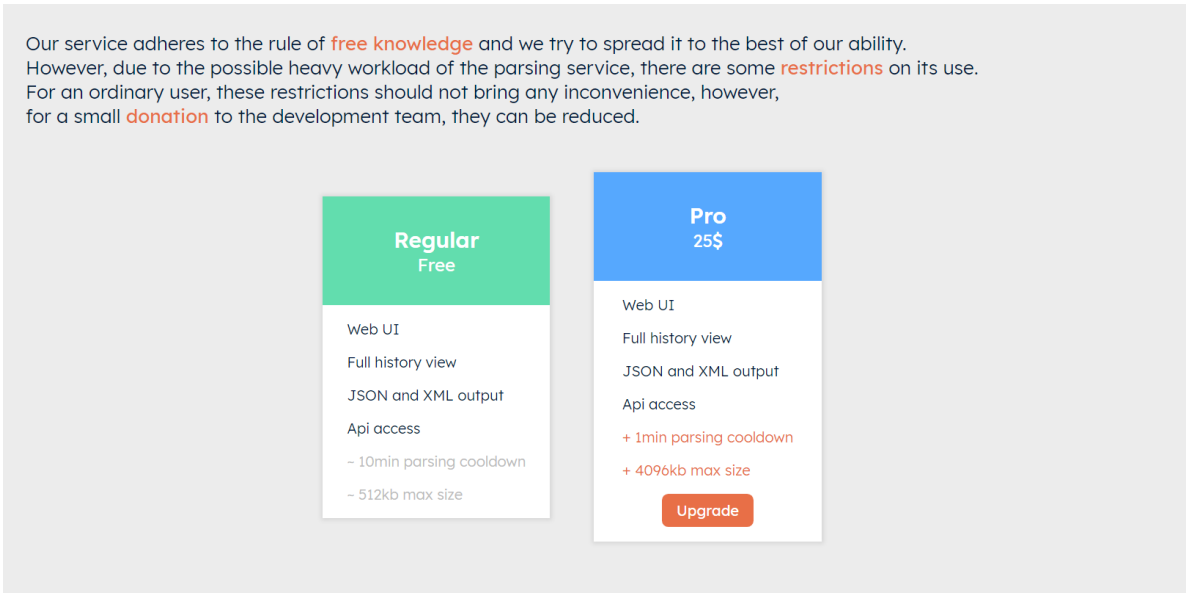

Рисунок 6.7 – Окно подписки

В данной секции можно увидеть типы аккаунтов, которые существуют в данном сервисе, а также просмотреть их ограничения. Как видно на рисунке 6.7, оба аккаунта отличаются лишь максимальной частотой запросов и максимальным размером файла. Для аккаунта типа «Pro» данный ограничения соответствуют одному запросу в минуту и 4 Мбайта данных. Для смены типа аккаунта необходимо нажать кнопку «Upgrade».

6.2.3.4 Документация Api

Для доступа к этой секции необходимо выбрать пункт меню «Api».

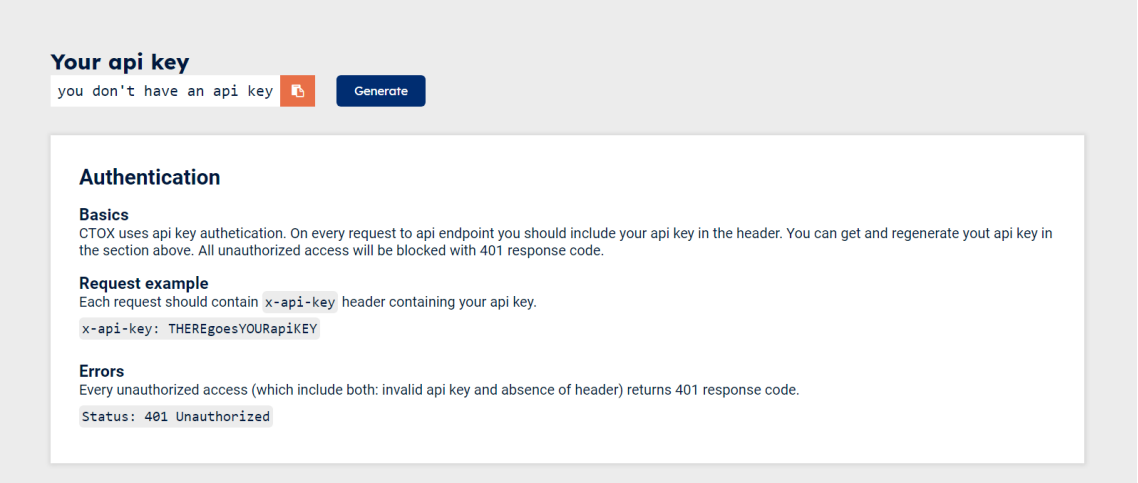

Рисунок 6.8 – Окно документации

В данной секции можно просмотреть информацию об авторизации по ключу, конечных точках, их параметрах и типах. Для получения ключа, необходимого для доступа к Api, следует нажать кнопку «Generate» и скопировать ключ в буфер обмена. Внимание! Данный ключ является авторизацинным, его использование влечет за собой применение всех ограничений на сервис, его следует хранить в месте, не доступным для публики.

ЗАКЛЮЧЕНИЕ

В ходе работы над курсовой работой были проанализированы системы авторизации и аутентификации, СУБД, веб-фреймворк Asp.Net Core и движок генерации HTML разметок, исследованы системы контейнеризации, общие принципы теории трансляторов, а также утилиты Flex и Bison.

В процессе разработки были исследованы существующие аналоги. Результатом этого анализа явилось обобщение преимуществ и недостатков существующих решений, которые были учтены при разработке функциональных требований данного проекта.

На основе функциональных требований было произведено проектирование программного средства, с учетом корректности выходных данных, а также защитой уязвимой информации.

В программном средстве реализованы функции для обработки всех запросов пользователей, правильного и интуитивно понятного отображения информации, а также взаимодействия нескольких систем воедино.

Согласно функциональным требованиям были разработаны тестовые сценарии, которые были успешно пройдены в ходе тестирования программного средства.

На завершающем этапе подробно описано руководство использования программного средства, которое позволяет легко освоить работу с программным средством.

Главной целью данного проекта было предоставление удобного способа синтаксического анализа исходных кодов, посредством API запросов. Эта цель была успешно достигнута. При этом программное средство имеет потенциал для улучшения. В качестве дальнейших совершенствований можно выделить следующие моменты:

- отзывчивый дизайн графического интерфейса, пригодный для удобного использования на экранах с меньшим размером;
- панель администрирования аккаунтов пользователей и их запросов;
- пункт с настройками аккаунта (таких как смена пароля и почты);
- возможное генерирование блок-схем на основе полученного абстрактного синтаксического дерева.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Словарь компьютерный терминов [Электронный ресурс]. – Режим доступа: https://computer.slovaronline.com/761-VEB-SERVIS.

[2] The free dictionary [Электронный ресурс]. – Режим доступа: https://encyclopedia2.thefreedictionary.com/Web+services.

[3] Random.org [Электронный ресурс]. – Режим доступа: https://www.random.org/randomness/.

[4] Architectural Styles and the Design of Network-based Software Architectures [Электронный ресурс]. – Режим доступа: https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm.

[5] История комьютера [Электронный ресурс]. – Режим доступа: http://chernykh.net/content/view/221/234/.

[6] Dribbble [Электронный ресурс]. – Режим доступа: https://dribbble.com/.

[7] Google Cloud [Электронный ресурс]. – Режим доступа: https://console.cloud.google.com/.

[8] ANSI C Grammar [Электронный ресурс]. – Режим доступа: https://www.lysator.liu.se/c/ANSI-C-grammar-y.html.

[9] Стандарт предприятия. Дипломные проекты. Общие требования [Электронный ресурс]. – Режим доступа: https://www.bsuir.by/m/12_100229_1_80040.pdf.

ПРИЛОЖЕНИЕ А. ИСХОДНЫЙ КОД ПРОГРАММЫ

```
using System;
using System.Diagnostics;
using System.IO;
using System.Threading.Tasks;

namespace CtoxService
{
    public class Ctox
    {
        private readonly string command;

        public Ctox(string command)
        {
            if (!File.Exists(command))
            {
                throw new ArgumentException("File doesn't exist. Check if
it's compiled.");
            }

            this.command = command;
        }

        public string Parse(string content)
        {
            // Создание процесса, с перенаправленными стандартными потоками
            using var process = new Process();
            process.StartInfo.FileName = command;
            process.StartInfo.RedirectStandardInput = true;
            process.StartInfo.RedirectStandardOutput = true;

            // Запуск процесса и запись в поток ввода
            process.Start();
            process.StandardInput.Write(content);

            // Закрытие потока ввода
            process.StandardInput.Close();

            // Закрытие процесса и сохранение вывода
            var result = process.StandardOutput.ReadToEnd();
            process.WaitForExit();

            return result;
        }

        public Task<string> ParseAsync(string content)
        {
            return Task.Run(() => Parse(content));
        }
    }
}

using System.Threading;
using Microsoft.AspNetCore.Builder;
```

```
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.DependencyInjection;

namespace CtoxWebApp
{
    public static class ApplicationBuilderExtensions
    {
        private const int MaxRetries = 10;
        private const int RetryDelay = 5000;

        public static void EnsureConnected<T>(this IApplicationBuilder app)
            where T : DbContext
        {
            using var scope = app.ApplicationServices.CreateScope();
            var context = (T) scope.ServiceProvider.GetService(typeof(T));

            for (int i = 0; i < MaxRetries; i++)
            {
                if (context.IsConnected())
                {
                    return;
                }
                Thread.Sleep(RetryDelay);
            }

            throw new System.InvalidOperationException("Could not establish
connection");
        }

        public static void EnsureMigrated<T>(this IApplicationBuilder app)
            where T : DbContext
        {
            using var scope = app.ApplicationServices.CreateScope();
            var context = (T) scope.ServiceProvider.GetService(typeof(T));
            context.Database.Migrate();
        }

        private static bool IsConnected(this DbContext context)
        {
            try
            {
                _ = context.Database.ExecuteSqlRaw("SELECT 1");
                return true;
            }
            catch
            {
                return false;
            }
        }
    }
}

using System;

namespace CtoxWebApp.Attributes.Filters
{
    public class AllowNoApi : Attribute
```

```
    {
    }
}

using System;
using System.Data.Entity;
using System.Linq;
using CtoxWebApp.DAL;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Controllers;
using Microsoft.AspNetCore.Mvc.Filters;

namespace CtoxWebApp.Attributes.Filters
{
    public class ApiKey : Attribute, IActionFilter
    {
        public ApiKey()
        {
        }

        public string Roles { get; set; }

        public void OnActionExecuting(ActionExecutingContext context)
        {
            // ReSharper disable once PossibleNullReferenceException
            if ((context.ActionDescriptor as ControllerActionDescriptor)
                .MethodInfo
                .GetCustomAttributes(typeof(AllowNoApi), false)
                .FirstOrDefault() != null)
            {
                return;
            }

            if (!context.HttpContext.Request.Headers.ContainsKey("x-api-
key"))
            {
                context.Result = new UnauthorizedObjectResult("No api key");
                return;
            }

            var dbContext =
(AppDbContext)context.HttpContext.RequestServices.GetService(typeof(AppDbCont
ext));

            var api = dbContext.Apis
                .Include(a => a.User)
                .FirstOrDefault(a =>
a.Key.Equals(context.HttpContext.Request.Headers["x-api-key"]));

            if (api is null)
            {
                context.Result = new UnauthorizedObjectResult("No api key");
                return;
            }

            if (Roles is null || Roles.Length == 0)
            {
```

```
                return;
            }

            foreach (var role in Roles.Split(','))
            {
                if (api.User.Role.ToString().Equals(role))
                {
                    return;
                }
            }

            context.Result = new UnauthorizedObjectResult("No api key");
        }

        public void OnActionExecuted(ActionExecutedContext context)
        {
        }
    }
}

#nullable enable
using System.ComponentModel.DataAnnotations;
using MimeKit;

namespace CtoxWebApp.Attributes.Validations
{
    public class EmailAttribute : ValidationAttribute
    {
        public override bool IsValid(object? value)
        {
            if (!(value is string vs))
            {
                return false;
            }

            return MailboxAddress.TryParse(vs, out MailboxAddress _);
        }
    }
}

#nullable enable
using System;
using System.ComponentModel.DataAnnotations;

namespace CtoxWebApp.Attributes.Validations
{
    [Flags]
    public enum PasswordHas
    {
        None = 0,
        Uppercase = 1,
        Lowercase = 2,
        Digit = 4,
        Special = 8,
    }

    public class StrongPasswordAttribute : ValidationAttribute
```

```
    {
        private readonly PasswordHas props;

        public StrongPasswordAttribute(PasswordHas props)
        {
            this.props = props;
        }

        public override bool IsValid(object? value)
        {
            if (value is null)
            {
                return false;
            }

            var vs = value.ToString();
            if (vs is null)
            {
                return false;
            }

            return (Score(vs) & props) == props;
        }

        private PasswordHas Score(string value)
        {
            PasswordHas result = PasswordHas.None;
            foreach (var c in value)
            {
                if (!result.HasFlag(PasswordHas.Uppercase) &&
char.IsUpper(c))
                {
                    result |= PasswordHas.Uppercase;
                    continue;
                }

                if (!result.HasFlag(PasswordHas.Lowercase) &&
char.IsLower(c))
                {
                    result |= PasswordHas.Lowercase;
                    continue;
                }

                if (!result.HasFlag(PasswordHas.Digit) && char.IsDigit(c))
                {
                    result |= PasswordHas.Digit;
                    continue;
                }

                if (!result.HasFlag(PasswordHas.Special) && char.IsSymbol(c))
                {
                    result |= PasswordHas.Special;
                }
            }

            return result;
        }
```

```
    }
}

#nullable enable
using System;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using CtoxWebApp.DAL;
using Microsoft.Extensions.DependencyInjection;

namespace CtoxWebApp.Attributes.Validations
{
    public class UniqueEmailAttribute : ValidationAttribute
    {
        protected override ValidationResult? IsValid(object? value,
ValidationContext validationContext)
        {
            if (value is null)
            {
                return new ValidationResult(null);
            }

            var context = validationContext.GetService<AppDbContext>();
            var vs = value.ToString();

            if (context?.Users.FirstOrDefault(u => u.Email.Equals(vs,
StringComparison.Ordinal)) != null)
            {
                return new ValidationResult(null);
            }

            return ValidationResult.Success;
        }
    }
}

#nullable enable
using System;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using CtoxWebApp.DAL;
using Microsoft.Extensions.DependencyInjection;

namespace CtoxWebApp.Attributes.Validations
{
    public class UniqueUsernameAttribute : ValidationAttribute
    {
        protected override ValidationResult? IsValid(object? value,
ValidationContext validationContext)
        {
            if (value is null)
            {
                return new ValidationResult(null);
            }

            var context = validationContext.GetService<AppDbContext>();
            var vs = value.ToString();
```

```
            if (context?.Users.FirstOrDefault(u => u.Username.Equals(vs,
StringComparison.Ordinal)) != null)
            {
                return new ValidationResult(null);
            }

            return ValidationResult.Success;
        }
    }
}

using System;
using System.Data.Entity;
using System.Linq;
using System.Threading.Tasks;
using CtoxWebApp.Attributes.Filters;
using CtoxWebApp.DAL;
using CtoxWebApp.Models.UserModel.Domain;
using CtoxWebApp.Services.Interfaces;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using Newtonsoft.Json;

namespace CtoxWebApp.Controllers
{
    [ApiKey(Roles = "Admin")]
    [Route("/api/admin")]
    public class AdminApiController : ControllerBase
    {
        private const string ContentTypeJson = "application/json";
        private readonly AppDbContext context;

        public AdminApiController(AppDbContext context)
        {
            this.context = context;
        }

        [HttpGet("users")]
        public IActionResult GetUsers()
        {
            var users =
                from u in context.Users
                select new {
                    u.Id,
                    u.Username,
                    u.Email,
                    u.Confirmed,
                    Role = u.Role.ToString()
                    };

            return Content(JsonConvert.SerializeObject(users),
ContentTypeJson);
        }

        [HttpDelete("user")]
        public async Task<IActionResult> DeleteUser(int? id)
```

```
        {
            if (id is null)
            {
                return BadRequest();
            }

            var user = context.Users.FirstOrDefault(u => u.Id == id);
            if (user is null)
            {
                return NotFound();
            }

            context.Users.Remove(user);
            await context.SaveChangesAsync();
            return Ok();
        }

        [HttpGet("stat")]
        public IActionResult GetStat(int? id)
        {
            if (id is null)
            {
                return BadRequest();
            }

            var user = context.Users.FirstOrDefault(u => u.Id == id);
            if (user is null)
            {
                return NotFound();
            }

            var api = context.Apis.FirstOrDefault(a => a.UserId == id);
            var res = context.PasswordRestores.Where(r => r.UserId ==
id).Count();

            if (api is null)
            {
                return Content(JsonConvert.SerializeObject(new { HasKey =
false, Restores = res }), ContentTypeJson);
            }

            var result = new {
                HasKey = true,
                Key = api.Key,
                LastUsed = api.LastUsed,
                Conversions = context.Conversions.Where(c => c.UserId ==
id).Count(),
                Restores = res
            };
            return Content(JsonConvert.SerializeObject(result),
ContentTypeJson);
        }

        [HttpPut("confirmed")]
        public async Task<IActionResult> SetConfirmed(int? id, bool?
confirmed)
        {
```

```
        if (id is null || confirmed is null)
        {
            return BadRequest();
        }

        var user = context.Users.FirstOrDefault(u => u.Id == id);
        if (user is null)
        {
            return NotFound();
        }

        user.Confirmed = (bool) confirmed;
        await context.SaveChangesAsync();

        return Ok();
    }

    [HttpPut("role")]
    public async Task<IActionResult> SetRole(int? id, Role? role)
    {
        if (id is null || role is null)
        {
            return BadRequest();
        }

        if (!Enum.IsDefined(typeof(Role), role))
        {
            return BadRequest();
        }

        var user = context.Users.FirstOrDefault(u => u.Id == id);
        if (user is null)
        {
            return NotFound();
        }

        user.Role = (Role) role;
        await context.SaveChangesAsync();

        return Ok();
    }

    [HttpGet("conversions")]
    public IActionResult GetConversions(int? id)
    {
        if (id is null)
        {
            return BadRequest();
        }

        var conversions = context.Conversions
            .Where(c => c.UserId == id)
            .Select(c => new { c.Id, c.Time, Type = c.Type.ToString()});

        if (conversions.Count() == 0)
        {
            return NotFound();
```

```
            }

            return Content(JsonConvert.SerializeObject(conversions),
ContentTypeJson);
        }

        [HttpDelete("conversion")]
        public async Task<IActionResult> DeleteConversion(int? id)
        {
            if (id is null)
            {
                return BadRequest();
            }

            var conversion = context.Conversions.FirstOrDefault(c => c.Id ==
id);
            if (conversion is null)
            {
                return NotFound();
            }

            context.Conversions.Remove(conversion);
            await context.SaveChangesAsync();
            return Ok();
        }

        [HttpGet("view")]
        public IActionResult ViewConversion(int? id, [FromServices]
IStringCompressService compress)
        {
            if (id is null)
            {
                return BadRequest();
            }

            var conversion = context.Conversions.FirstOrDefault(c => c.Id ==
id);
            if (conversion is null)
            {
                return NotFound();
            }

            var result = new {
                Initial = compress.Decompress(conversion.Initial),
                Result = compress.Decompress(conversion.Result)
            };

            return Content(JsonConvert.SerializeObject(result),
ContentTypeJson);
        }
    }
}

using System;
using System.Data.Entity;
using System.Linq;
using System.Threading.Tasks;
```

```
using System.Xml.Linq;
using CtoxWebApp.Attributes.Filters;
using CtoxWebApp.DAL;
using CtoxWebApp.Models.ApiModel.Domain;
using CtoxWebApp.Services.Implementations;
using CtoxWebApp.Services.Interfaces;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using Newtonsoft.Json;

namespace CtoxWebApp.Controllers
{
    [ApiKey]
    [Route("api")]
    public class ApiController : ControllerBase
    {
        private const string TestResultText = "<function>" +
                                              "<prototype> void test ( )
</prototype>" +
                                              "<body>" +
                                              "<expression hascalls =
\"true\">" +
                                              "<funccall>" +
                                              "<name> printf </name>" +
                                              "<arguments>\"%s\\n\" , \"Hello
from api!\" </arguments>" +
                                              "</funccall>" +
                                              "<text> printf ( \"%s\\n\" ,
\"Hello from api!\" ) </text>" +
                                              "</expression>" +
                                              "</body>" +
                                              "</function>";

        private const string ContentTypeJson = "application/json";
        private const string ContentTypeXml = "application/xml";
        private const string ApiKeyHeader = "x-api-key";

        private static readonly string TestResultJson =
JsonConvert.SerializeXNode(XElement.Parse(TestResultText));

        private readonly AppDbContext context;
        private readonly RestrictionService restriction;
        private readonly IParseAsyncService parseAsync;
        private readonly IStringCompressService compress;

        public ApiController(AppDbContext context, RestrictionService
restriction, IParseAsyncService parseAsync,
            IStringCompressService compress)
        {
            this.context = context;
            this.restriction = restriction;
            this.parseAsync = parseAsync;
            this.compress = compress;
        }

        [HttpGet("test")]
        public IActionResult Test(bool? json)
```

```
        {
            if (json != true)
            {
                return Content(TestResultText, ContentTypeXml);
            }

            return Content(TestResultJson, ContentTypeJson);
        }

        [Authorize]
        [AllowNoApi]
        [HttpGet("create")]
        public async Task<IActionResult> Create([FromServices] IHashService
hash)
        {
            var user = context.Users.First(u =>
u.Username.Equals(HttpContext.User.Identity.Name));
            var api = context.Apis.FirstOrDefault(a => a.UserId == user.Id);

            if (api is null)
            {
                api = new Api
                {
                    Key = hash.GetRandom(),
                    LastUsed = DateTime.MinValue,
                    UserId = user.Id
                };

                context.Apis.Add(api);
            }
            else
            {
                api.Key = hash.GetRandom();
            }

            await context.SaveChangesAsync();
            return StatusCode(201, new {api.Key});
        }

        [HttpPost("parse")]
        public async Task<IActionResult> Parse(
            bool? json,
            [FromBody] ParseRequest request,
            [FromHeader(Name = ApiKeyHeader)] string key)
        {
            if (request is null
                || string.IsNullOrWhiteSpace(request.Data))
            {
                return BadRequest("Empty");
            }

            var api = context.Apis
                .Include(a => a.User)
                .First(a => a.Key.Equals(key));

            if (!restriction.IsAllowedTimeout(api))
            {
```

```
                return StatusCode(429, "Timeout");
            }

            if (!restriction.IsAllowedSize(api, request.Data.Length))
            {
                return StatusCode(413, "Size");
            }

            api.LastUsed = DateTime.Now;

            var parseResult = await parseAsync.ParseAsync(request.Data);
            if (json == true)
            {
                parseResult =
JsonConvert.SerializeXNode(XElement.Parse(parseResult));
            }

            context.Conversions.Add(new Conversion
            {
                Initial = compress.Compress(request.Data),
                Result = compress.Compress(parseResult),
                Type = json == true ? ParseType.Json : ParseType.Xml,
                Time = api.LastUsed,
                UserId = api.UserId,
            });
            await context.SaveChangesAsync();

            return Content(parseResult, json == true ? ContentTypeJson :
ContentTypeXml);
        }

        [HttpGet("history")]
        public IActionResult History(
            DateTime? day,
            bool? ascending,
            int? skip,
            int? limit,
            [FromHeader(Name = ApiKeyHeader)] string key)
        {
            var api = context.Apis
                .Include(a => a.User)
                .First(a => a.Key.Equals(key));

            var conversions = context.Conversions
                .Where(c => c.UserId == api.UserId);

            conversions = SearchConversions(conversions, day, ascending,
skip, limit);

            var data = conversions.Select(c => new
            {
                c.Id,
                c.Time,
                c.Type
            });

            var result = new {Amount = data.Count(), Data = data};
```

```
            return Content(JsonConvert.SerializeObject(result),
ContentTypeJson);
        }

        [HttpGet("view")]
        public IActionResult View(int? id, [FromHeader(Name = ApiKeyHeader)]
string key)
        {
            if (id is null)
            {
                return BadRequest();
            }

            var api = context.Apis
                .Include(a => a.User)
                .First(a => a.Key.Equals(key));

            var conversion = context.Conversions.FirstOrDefault(c => c.Id ==
id);

            if (conversion is null || conversion.UserId != api.UserId)
            {
                return NotFound();
            }

            var result = new
            {
                Initial = compress.Decompress(conversion.Initial),
                Result = compress.Decompress(conversion.Result)
            };

            return Content(JsonConvert.SerializeObject(result),
ContentTypeJson);
        }

        private IQueryable<Conversion> SearchConversions(
            IQueryable<Conversion> conversions,
            DateTime? day,
            bool? ascending,
            int? skip,
            int? limit)
        {
            if (day != null)
            {
                conversions = conversions.Where(c => c.Time.Date ==
((DateTime) day).Date);
            }

            conversions = ascending == false ?
                conversions.OrderByDescending(c => c.Time) :
                conversions.OrderBy(c => c.Time);

            if (skip != null)
            {
                conversions = conversions.Skip((int) skip);
            }
```

```
            if (limit != null)
            {
                conversions = conversions.Take((int) limit);
            }

            return conversions;
        }
    }
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Security.Claims;
using System.Threading.Tasks;
using CtoxWebApp.DAL;
using CtoxWebApp.Models.UserModel.Domain;
using CtoxWebApp.Models.UserModel.View;
using CtoxWebApp.Services.Interfaces;
using Microsoft.AspNetCore.Authentication;
using Microsoft.AspNetCore.Authentication.Cookies;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;

namespace CtoxWebApp.Controllers
{
    public class AuthController : Controller
    {
        private const string LoginNotFilledErrorMessage =
            "Both username and password should be filled in. Please check
your username and password and try again.";

        private const string LoginNotFoundErrorMessage =
            "We couldn't find an account matching the username and password
you entered. Please check your username and password and try again.";

        private const string LoginNotConfirmedErrorMessage =
            "It seems like you didn't confirm your email address. Please make
sure that you followed the link sent to your email.";

        private const string VerificationInvalidErrorMessage =
            "Invalid verification string. Please make sure you followed the
link correctly.";

        private const string VerificationAgainErrorMessage = "Your email have
been already confirmed.";
        private const string VerificationVerifiedInfoMessage = "Your email
has been verified.";

        private const string RestoreInfoMessage =
            "If this email address was used to create an account,
instructions to reset your password will be sent to you. Please check your
email.";

        private const string ResetSuccessInfoMessage = "Password for
specified user was successfully reset.";
```

```
        private readonly AppDbContext dbContext;
        private readonly IHashService hashService;
        private readonly IEmailSenderService sender;

        public AuthController(AppDbContext dbContext, IHashService
hashService, IEmailSenderService sender)
        {
            this.dbContext = dbContext;
            this.hashService = hashService;
            this.sender = sender;
        }

        public IActionResult Login()
        {
            if (User.Identity.IsAuthenticated)
            {
                return RedirectToAction("Index", "Home");
            }

            return View();
        }

        [HttpPost]
        public async Task<IActionResult> Login(UserLogin user)
        {
            if (user.Username is null || user.Password is null)
            {
                ViewData["error-message"] =
                    LoginNotFilledErrorMessage;
                return View();
            }

            var hash = hashService.GetHash(string.Concat(user.Username,
hashService.GetHash(user.Password)));

            var result = dbContext.Users
                .FirstOrDefault(u => u.Username.Equals(user.Username,
StringComparison.Ordinal) &&
                                     u.PasswordHash.Equals(hash,
StringComparison.Ordinal));

            if (result is null)
            {
                ViewData["error-message"] =
                    LoginNotFoundErrorMessage;
                return View();
            }

            if (!result.Confirmed)
            {
                ViewData["error-message"] =
                    LoginNotConfirmedErrorMessage;
                return View();
            }

            await Authenticate(result);
            return RedirectToAction("Index", "Home");
```

```
        }

        public IActionResult Register()
        {
            return View();
        }

        [HttpPost]
        public async Task<IActionResult> Register(UserRegister user)
        {
            if (!ModelState.IsValid) return View(user);

            var registered = new User
            {
                Username = user.Username,
                Email = user.Email,
                PasswordHash =
                    hashService.GetHash(string.Concat(user.Username,
hashService.GetHash(user.Password))),
                Confirmed = false,
                Role = Role.Regular,
            };

            dbContext.Users.Add(registered);
            await dbContext.SaveChangesAsync();

            if (User.Identity.IsAuthenticated)
            {
                await
HttpContext.SignOutAsync(CookieAuthenticationDefaults.AuthenticationScheme);
            }

            await SendVerification(registered);

            return Redirect("Login");
        }

        public async Task<IActionResult> Verify(string verificationString)
        {
            if (string.IsNullOrWhiteSpace(verificationString))
            {
                return BadRequest();
            }

            if (User.Identity.IsAuthenticated)
            {
                await HttpContext.SignOutAsync();
            }

            var result = dbContext.UserVerifications
                .Include(v => v.User)
                .FirstOrDefault(v =>
v.Verification.Equals(verificationString, StringComparison.Ordinal));

            if (result is null)
            {
                ViewData["error-message"] =
```

```
                    VerificationInvalidErrorMessage;
                return View("Login");
            }

            if (result.User.Confirmed)
            {
                ViewData["error-message"] =
                    VerificationAgainErrorMessage;
                return View("Login");
            }

            result.User.Confirmed = true;
            await dbContext.SaveChangesAsync();

            ViewData["info-message"] = VerificationVerifiedInfoMessage;
            return View("Login");
        }

        public IActionResult Restore()
        {
            return View();
        }

        [HttpPost]
        public async Task<IActionResult> Restore(UserRestoreRequest user)
        {
            var result = dbContext.Users.FirstOrDefault(u =>
u.Email.Equals(user.Email));
            if (result != null)
            {
                var hash = hashService.GetRandom();
                dbContext.PasswordRestores.Add(new PasswordRestore
                {
                    UserId = result.Id,
                    Restore = hash,
                    Valid = true,
                });
                await dbContext.SaveChangesAsync();
                await sender.SendEmailAsync(result.Username, result.Email,
                    $"Someone tries to reset your password on CTOX.\nIf it
were you, please follow the link https://localhost:5001/Reset/{hash}");
            }

            ViewData["info-message"] = RestoreInfoMessage;
            return View("Login");
        }

        public async Task<IActionResult> Reset(string resetString)
        {
            if (string.IsNullOrWhiteSpace(resetString))
            {
                return BadRequest();
            }

            if (User.Identity.IsAuthenticated)
            {
                await HttpContext.SignOutAsync();
```

```
            }

            var result = dbContext.PasswordRestores
                .Include(p => p.User)
                .FirstOrDefault(p => p.Restore.Equals(resetString));

            if (result is null || !result.Valid)
            {
                return BadRequest();
            }

            return View(new UserRestore {Restore = resetString});
        }

        [HttpPost]
        public async Task<IActionResult> Reset(UserRestore user)
        {
            if (!ModelState.IsValid) return View(user);
            var result = dbContext.PasswordRestores
                .Include(p => p.User)
                .FirstOrDefault(p => p.Restore.Equals(user.Restore));

            if (result is null || !result.Valid)
            {
                return BadRequest();
            }

            result.Valid = false;
            result.User.PasswordHash =
hashService.GetHash(string.Concat(result.User.Username,
hashService.GetHash(user.Password)));
            await dbContext.SaveChangesAsync();

            ViewData["info-message"] = ResetSuccessInfoMessage;
            return View("Login");
        }

        public async Task<IActionResult> Logout()
        {
            await
HttpContext.SignOutAsync(CookieAuthenticationDefaults.AuthenticationScheme);
            return Redirect("Login");
        }

        private Task Authenticate(User user)
        {
            var claims = new List<Claim>
            {
                new Claim(ClaimsIdentity.DefaultNameClaimType,
user.Username),
                new Claim(ClaimsIdentity.DefaultRoleClaimType,
user.Role.ToString()),
            };

            var id = new ClaimsIdentity(claims, "ApplicationCookie",
ClaimsIdentity.DefaultNameClaimType,
                ClaimsIdentity.DefaultRoleClaimType);
```

```
            return
HttpContext.SignInAsync(CookieAuthenticationDefaults.AuthenticationScheme,
new ClaimsPrincipal(id));
        }

        private async Task SendVerification(User user)
        {
            var verification = hashService.GetRandom();
            dbContext.UserVerifications.Add(new UserVerification
            {
                UserId = user.Id,
                Verification = verification,
            });
            await dbContext.SaveChangesAsync();
            await sender.SendEmailAsync(user.Username, user.Email,
                $"To verify your email address on CTOX, please follow the
link.\nhttps://localhost:5001/Verify/{verification}");
        }
    }
}

using System.Collections.Generic;
using CtoxWebApp.Models;
using Microsoft.AspNetCore.Mvc;

namespace CtoxWebApp.Controllers
{
    public class ErrorController : Controller
    {
        private readonly Dictionary<int, ErrorInfo> Pages = new
Dictionary<int, ErrorInfo>()
        {
            { 401, new ErrorInfo { Code = 401, Title = "Unauthorized.",
Description = "Seems like you're trying to access something, that's not set
up for anons. Authenticate and try again!", RedirectLink = "/Home/Index" } },
            { 404, new ErrorInfo { Code = 404, Title = "Page not found.",
Description = "You must have picked the wrong path, beacause I cant find
anything useful!", RedirectLink = "/" } }
        };

        [HttpGet("error")]
        public IActionResult Error(int? code)
        {
            if (code != null)
            {
                HttpContext.Response.StatusCode = code.Value;
            }

            if (!Pages.ContainsKey(code.Value))
            {
                return new EmptyResult();
            }

            return View(Pages[code.Value]);
        }
    }
```

```
}

using System.Data.Entity;
using System.Linq;
using System.Security.Claims;
using System.Threading.Tasks;
using CtoxWebApp.DAL;
using CtoxWebApp.Extensions;
using CtoxWebApp.Models.ApiModel.Domain;
using CtoxWebApp.Models.ApiModel.View;
using CtoxWebApp.Models.UserModel.Domain;
using CtoxWebApp.Services.Implementations;
using Microsoft.AspNetCore.Authentication;
using Microsoft.AspNetCore.Authentication.Cookies;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;

namespace CtoxWebApp.Controllers
{
    [Authorize]
    public class HomeController : Controller
    {
        private const int HistoryMaxAmount = 10;
        private readonly AppDbContext context;

        public HomeController(AppDbContext context)
        {
            this.context = context;
        }

        public IActionResult Index()
        {
            return View("Convert");
        }

        [HttpPost]
        public async Task<IActionResult> Index(ParseRequestUi request,
[FromServices] ApiController apiController)
        {
            if (request is null
                || string.IsNullOrWhiteSpace(request.Data))
            {
                return BadRequest("Empty");
            }

            var api = context.Apis
                .Include(a => a.User)
                .FirstOrDefault(a =>
a.User.Username.Equals(User.Identity.Name));

            if (api is null)
            {
                return Unauthorized("No api key");
            }

            var result = await apiController.Parse(
                request.Type == ParseType.Json,
```

```
                new ParseRequest
                {
                    Data = request.Data
                }, api.Key);

            return result;
        }

        public IActionResult History()
        {
            var api = context.Apis
                .Include(a => a.User)
                .FirstOrDefault(a =>
a.User.Username.Equals(User.Identity.Name));

            if (api is null)
            {
                return View(new ConversionUi
                {
                    HasKey = false,
                    Amount = 0,
                    Conversions = Enumerable.Empty<Conversion>()
                });
            }

            var conversions = context.Conversions
                .Where(c => c.UserId == api.UserId)
                .OrderByDescending(c => c.Time)
                .Take(HistoryMaxAmount)
                .ToList();

            return View(new ConversionUi
            {
                HasKey = true,
                Key = api.Key,
                Amount = conversions.Count,
                Conversions = conversions
            });
        }

        public IActionResult Subscription([FromServices] RestrictionService
restriction)
        {
            ViewData["regular-timeout"] = restriction.RegularTimeout;
            ViewData["super-timeout"] = restriction.SuperTimeout;
            ViewData["regular-size"] = restriction.RegularSize;
            ViewData["super-size"] = restriction.SuperSize;
            return View();
        }

        public async Task<IActionResult> Unsubscribe()
        {
            var user = context.Users.First(u =>
u.Username.Equals(User.Identity.Name));
            if (user.Role != Role.Admin)
            {
                user.Role = Role.Regular;
```

```
                var id = User;

((ClaimsIdentity)User.Identity).UpdateClaim(ClaimsIdentity.DefaultRoleClaimTy
pe, Role.Regular.ToString());
                await HttpContext.SignOutAsync();
                await
HttpContext.SignInAsync(CookieAuthenticationDefaults.AuthenticationScheme,
id);
                await context.SaveChangesAsync();
            }

            return RedirectToAction("Subscription");
        }

        public async Task<IActionResult> Upgrade()
        {
            var user = context.Users.First(u =>
u.Username.Equals(User.Identity.Name));
            if (user.Role == Role.Regular)
            {
                user.Role = Role.Super;
                var id = User;

((ClaimsIdentity)User.Identity).UpdateClaim(ClaimsIdentity.DefaultRoleClaimTy
pe, Role.Super.ToString());
                await HttpContext.SignOutAsync();
                await
HttpContext.SignInAsync(CookieAuthenticationDefaults.AuthenticationScheme,
id);
                await context.SaveChangesAsync();
            }

            return RedirectToAction("Subscription");
        }

        public IActionResult Api()
        {
            var api = context.Apis
                .Include(a => a.User)
                .FirstOrDefault(a =>
a.User.Username.Equals(User.Identity.Name));

            ViewData["api-key"] = api is null ? string.Empty : api.Key;
            return View();
        }
    }
}

using CtoxWebApp.Models.ApiModel.Domain;
using CtoxWebApp.Models.UserModel.Domain;
using Microsoft.EntityFrameworkCore;

namespace CtoxWebApp.DAL
{
    public class AppDbContext : DbContext
    {
        public AppDbContext(DbContextOptions<AppDbContext> options)
```

```
            : base(options)
        {
        }

        public DbSet<User> Users { get; set; }
        public DbSet<UserVerification> UserVerifications { get; set; }
        public DbSet<PasswordRestore> PasswordRestores { get; set; }
        public DbSet<Api> Apis { get; set; }
        public DbSet<Conversion> Conversions { get; set; }
    }
}


using System;
using System.Linq;
using System.Security.Claims;

namespace CtoxWebApp.Extensions
{
    public static class ClaimsIdentityExtensions
    {
        public static void UpdateClaim(this ClaimsIdentity identity, string
type, string value)
        {
            var claim = identity.Claims.FirstOrDefault(c =>
c.Type.Equals(type));
            if (claim is null)
            {
                throw new InvalidOperationException("Identity doesnt have
specified type");
            }

            identity.RemoveClaim(claim);
            identity.AddClaim(new Claim(type, value));
        }
    }
}

using System;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc.Rendering;
using Microsoft.AspNetCore.Mvc.TagHelpers;
using Microsoft.AspNetCore.Mvc.ViewFeatures;
using Microsoft.AspNetCore.Razor.TagHelpers;

namespace CtoxWebApp.Helpers.Tag
{
    [HtmlTargetElement("a", Attributes = "active-title")]
    [HtmlTargetElement("a", Attributes = "active-class")]
    public class RouterTagHelper : TagHelper
    {
        [ViewContext]
        [HtmlAttributeNotBound]
        public ViewContext ViewContext { get; set; }

        [HtmlAttributeName("active-title")]
        public string Title { get; set; }
```

```
        [HtmlAttributeName("active-class")]
        public string Active { get; set; }

        public override Task ProcessAsync(TagHelperContext context,
TagHelperOutput output)
        {
            if (Title is null && !ViewContext.ViewData.ContainsKey("title"))
            {
                throw new ArgumentException("View data should contain title
value if no title present");
            }

            var content = output.GetChildContentAsync().Result.GetContent();
            bool condition = Title?.Equals(content) ??
ViewContext.ViewData["title"].Equals(content);

            if (condition)
            {
                var tagBuilder = new TagBuilder("a");
                tagBuilder.AddCssClass(Active ?? "active");
                output.MergeAttributes(tagBuilder);
            }

            return base.ProcessAsync(context, output);
        }
    }
}

using System;
using CtoxWebApp.Models.UserModel.Domain;

namespace CtoxWebApp.Models.ApiModel.Domain
{
    public class Api
    {
        public int Id { get; set; }
        public string Key { get; set; }
        public DateTime LastUsed { get; set; }
        public int UserId { get; set; }
        public virtual User User { get; set; }
    }
}

using System;
using CtoxWebApp.Models.UserModel.Domain;

namespace CtoxWebApp.Models.ApiModel.Domain
{
    public class Conversion
    {
        public int Id { get; set; }
        public DateTime Time { get; set; }
        public ParseType Type { get; set; }
        public string Initial { get; set; }
        public string Result { get; set; }
        public int UserId { get; set; }
```

```
        public virtual User User { get; set; }
    }
}

namespace CtoxWebApp.Models.ApiModel.Domain
{
    public class ParseRequest
    {
        public string Data { get; set; }
    }
}

namespace CtoxWebApp.Models.ApiModel.Domain
{
    public enum ParseType
    {
        Xml,
        Json
    }
}

using System.Collections.Generic;
using CtoxWebApp.Models.ApiModel.Domain;

namespace CtoxWebApp.Models.ApiModel.View
{
    public class ConversionUi
    {
        public bool HasKey { get; set; }
        public string Key { get; set; }
        public int Amount { get; set; }
        public IEnumerable<Conversion> Conversions { get; set; }
    }
}

using CtoxWebApp.Models.ApiModel.Domain;

namespace CtoxWebApp.Models.ApiModel.View
{
    public class ParseRequestUi
    {
        public string Data { get; set; }
        public ParseType Type { get; set; }
    }
}

namespace CtoxWebApp.Models
{
    public class ErrorInfo
    {
        public int Code { get; set; }
        public string Title { get; set; }
        public string Description { get; set; }
        public string RedirectLink { get; set; }
    }
}
```

```
namespace CtoxWebApp.Models.UserModel.Domain
{
    public class PasswordRestore
    {
        public int Id { get; set; }
        public string Restore { get; set; }
        public bool Valid { get; set; }
        public int UserId { get; set; }
        public virtual User User { get; set; }
    }
}

namespace CtoxWebApp.Models.UserModel.Domain
{
    public enum Role
    {
        Regular,
        Super,
        Admin,
    }
    public class User
    {
        public int Id { get; set; }
        public string Username { get; set; }
        public string Email { get; set; }
        public string PasswordHash { get; set; }
        public bool Confirmed { get; set; }
        public Role Role { get; set; }
    }
}

namespace CtoxWebApp.Models.UserModel.Domain
{
    public class UserVerification
    {
        public int Id { get; set; }
        public int UserId { get; set; }
        public virtual User User { get; set; }
        public string Verification { get; set; }
    }
}

using System.ComponentModel.DataAnnotations;

namespace CtoxWebApp.Models.UserModel.View
{
    public class UserLogin
    {
        public string Username { get; set; }

        [DataType(DataType.Password)]
        public string Password { get; set; }
    }
}

using System.ComponentModel.DataAnnotations;
using CtoxWebApp.Attributes.Validations;
```

```
namespace CtoxWebApp.Models.UserModel.View
{
    public class UserRegister
    {
        [Required(ErrorMessage = "Username should be filled")]
        [StringLength(60, MinimumLength = 3, ErrorMessage = "Username length
should be between 3 and 60")]
        [RegularExpression("[a-zA-Z]+", ErrorMessage = "Username should
consist only of letters")]
        [UniqueUsername(ErrorMessage = "Username is taken")]
        public string Username { get; set; }

        [Required(ErrorMessage = "Email should be filled")]
        [DataType(DataType.EmailAddress)]
        [Email(ErrorMessage = "Email address is not valid")]
        [UniqueEmail(ErrorMessage = "Email is taken")]
        public string Email { get; set; }

        [Required(ErrorMessage = "Password should be filled")]
        [DataType(DataType.Password)]
        [StringLength(500, MinimumLength = 6, ErrorMessage = "Password should
be at least 6 length")]
        [StrongPassword(PasswordHas.Digit | PasswordHas.Lowercase |
PasswordHas.Uppercase, ErrorMessage = "Password should contain a digit,
lowercase and uppercase letter")]
        public string Password { get; set; }
    }
}

using System.ComponentModel.DataAnnotations;
using CtoxWebApp.Attributes.Validations;
using Microsoft.AspNetCore.Mvc;

namespace CtoxWebApp.Models.UserModel.View
{
    public class UserRestore
    {
        [Required(ErrorMessage = "Password should be filled")]
        [DataType(DataType.Password)]
        [StringLength(500, MinimumLength = 6, ErrorMessage = "Password should
be at least 6 length")]
        [StrongPassword(PasswordHas.Digit | PasswordHas.Lowercase |
PasswordHas.Uppercase, ErrorMessage = "Password should contain a digit,
lowercase and uppercase letter")]
        public string Password { get; set; }

        [Display(Name = "Repeat password")]
        [DataType(DataType.Password)]
        [Compare("Password", ErrorMessage = "Passwords are not equal")]
        public string RepeatPassword { get; set; }

        [HiddenInput(DisplayValue = false)]
        public string Restore { get; set; }
    }
}
```

```
namespace CtoxWebApp.Models.UserModel.View
{
    public class UserRestoreRequest
    {
        public string Email { get; set; }
    }
}

using Microsoft.AspNetCore.Hosting;
using Microsoft.Extensions.Hosting;

namespace CtoxWebApp
{
    public class Program
    {
        public static void Main(string[] args)
        {
            CreateHostBuilder(args).Build().Run();
        }

        public static IHostBuilder CreateHostBuilder(string[] args) =>
            Host.CreateDefaultBuilder(args)
                .ConfigureWebHostDefaults(webBuilder =>
                {
                    webBuilder.UseStartup<Startup>();
                });
    }
}


using System.Text;
using System.Threading.Tasks;
using CtoxWebApp.Services.Interfaces;
using MailKit.Net.Smtp;
using Microsoft.Extensions.Configuration;
using MimeKit;

namespace CtoxWebApp.Services.Implementations
{
    public class EmailSenderService: IEmailSenderService
    {
        private readonly string email;
        private readonly string password;

        public EmailSenderService(IConfiguration configuration)
        {
            email = configuration["Email:Name"];
            password = configuration["Email:Password"];
        }

        public async Task SendEmailAsync(string receiverName, string
receiverEmail, string body)
        {

            var message = new MimeMessage();
            var from = new MailboxAddress(Encoding.UTF8, "Ctox", email);
```

```
            var to = new MailboxAddress(Encoding.UTF8, receiverName,
receiverEmail);
            message.From.Add(from);
            message.To.Add(to);
            message.Subject = "Ctox Service Notification.";

            var bodyBuilder = new BodyBuilder
            {
                TextBody = body,
            };
            message.Body = bodyBuilder.ToMessageBody();

            var smtp = new SmtpClient();
            await smtp.ConnectAsync("smtp.gmail.com", 465, true);

            await smtp.AuthenticateAsync(Encoding.UTF8, email, password);

            await smtp.SendAsync(message);
            await smtp.DisconnectAsync(true);
            smtp.Dispose();
        }
    }
}

using System;
using System.IO;
using System.IO.Compression;
using System.Text;
using CtoxWebApp.Services.Interfaces;

namespace CtoxWebApp.Services.Implementations
{
    public class GzipCompressService: IStringCompressService
    {
        public string Compress(string content)
        {
            string result;
            using (var so = new MemoryStream())
            {
                using (var gzip = new GZipStream(so,
CompressionMode.Compress))
                {
                    using (var si = new
MemoryStream(Encoding.UTF8.GetBytes(content)))
                    {
                        si.CopyTo(gzip);
                    }
                }

                result = Convert.ToBase64String(so.ToArray());
            }

            return result;
        }

        public string Decompress(string content)
        {
```

```
            string result;
            using (var si = new
MemoryStream(Convert.FromBase64String(content)))
            {
                using (var gzip = new GZipStream(si,
CompressionMode.Decompress))
                {
                    using (var so = new MemoryStream())
                    {
                        gzip.CopyTo(so);
                        result = Encoding.UTF8.GetString(so.ToArray());
                    }
                }
            }

            return result;
        }
    }
}

using System;
using System.Security.Cryptography;
using System.Text;
using CtoxWebApp.Services.Interfaces;

namespace CtoxWebApp.Services.Implementations
{
    public class HashService: IHashService
    {
        public string GetHash(string content)
        {
            var data = Encoding.UTF8.GetBytes(content);
            return GetHash(data);
        }

        public string GetHashSafe(string content)
        {
            var data = Encoding.UTF8.GetBytes(content);
            return GetHashSafe(data);
        }

        public string GetHash(byte[] data)
        {
            var sha = new SHA256CryptoServiceProvider();
            var enc = sha.ComputeHash(data);
            return Convert.ToBase64String(enc);
        }

        public string GetHashSafe(byte[] data)
        {
            var sha = new SHA256CryptoServiceProvider();
            var enc = sha.ComputeHash(data);
            return Convert.ToBase64String(enc).TrimEnd('=').Replace('+', '-
').Replace('/', '_');
        }

        public string GetRandom()
```

```
        {
            var data = new byte[8];
            var rnd = new Random();
            rnd.NextBytes(data);
            return GetHashSafe(data);
        }
    }
}

using System;
using System.Text;
using System.Threading.Tasks;
using CtoxWebApp.Services.Interfaces;

namespace CtoxWebApp.Services.Implementations.Mocks
{
    public class ParseAsyncMock : IParseAsyncService
    {
        private readonly IHashService hash;

        public ParseAsyncMock(IHashService hash)
        {
            this.hash = hash;
        }

        public Task<string> ParseAsync(string content)
        {
            return Task.Run(() => {
                var resultBuilder = new StringBuilder();

                for (int i = 0; i < new Random().Next(10, 20); i++)
                {
                    resultBuilder.Append($"<text>{hash.GetRandom()}</text>");
                }

                return $"<parse>{resultBuilder}</parse>";
            });
        }
    }
}

using System.Threading.Tasks;
using CtoxService;
using CtoxWebApp.Services.Interfaces;
using Microsoft.Extensions.Configuration;

namespace CtoxWebApp.Services.Implementations
{
    public class ParseAsyncService : IParseAsyncService
    {
        private readonly Ctox parse;

        public ParseAsyncService(IConfiguration configuration)
        {
            parse = new Ctox(configuration["ParseCommand"]);
        }
```

```
        public Task<string> ParseAsync(string content)
        {
            return parse.ParseAsync(content);
        }
    }
}

using System;
using CtoxWebApp.Models.ApiModel.Domain;
using CtoxWebApp.Models.UserModel.Domain;
using Microsoft.Extensions.Configuration;

namespace CtoxWebApp.Services.Implementations
{
    public class RestrictionService
    {
        private readonly TimeSpan[] timeouts;
        private readonly int[] maxSizes;

        public RestrictionService(IConfiguration configuration)
        {
            var names = Enum.GetNames(typeof(Role));
            timeouts = new TimeSpan[names.Length];
            maxSizes = new int[names.Length];

            for (int i = 0; i < names.Length; i++)
            {
                timeouts[i] =
TimeSpan.FromMinutes(Convert.ToDouble(configuration[$"Restrictions:Timeouts:{
names[i]}"]));
                maxSizes[i] =
Convert.ToInt32(configuration[$"Restrictions:Sizes:{names[i]}"]);
            }
        }

        public int RegularTimeout => timeouts[(int) Role.Regular].Minutes;
        public int SuperTimeout => timeouts[(int) Role.Super].Minutes;
        public int RegularSize => maxSizes[(int) Role.Regular];
        public int SuperSize => maxSizes[(int) Role.Super];

        public bool IsAllowedTimeout(Api api)
        {
            var diff = DateTime.Now - api.LastUsed;
            return diff > timeouts[(int) api.User.Role];
        }

        public bool IsAllowedSize(Api api, int size)
        {
            return size <= maxSizes[(int) api.User.Role];
        }
    }
}

using System.Threading.Tasks;

namespace CtoxWebApp.Services.Interfaces
{
```

```
    public interface IEmailSenderService
    {
        Task SendEmailAsync(string receiverName, string receiverEmail, string
body);
    }
}

namespace CtoxWebApp.Services.Interfaces
{
    public interface IHashService
    {
        public string GetHash(string content);
        public string GetHash(byte[] data);
        public string GetHashSafe(string content);
        public string GetHashSafe(byte[] data);
        public string GetRandom();
    }
}

using System.Threading.Tasks;

namespace CtoxWebApp.Services.Interfaces
{
    public interface IParseAsyncService
    {
        Task<string> ParseAsync(string content);
    }
}

namespace CtoxWebApp.Services.Interfaces
{
    public interface IStringCompressService
    {
        string Compress(string content);
        string Decompress(string content);
    }
}

using System.Threading.Tasks;
using CtoxWebApp.Attributes.Filters;
using CtoxWebApp.DAL;
using CtoxWebApp.Services.Implementations;
using CtoxWebApp.Services.Interfaces;
using Microsoft.AspNetCore.Authentication.Cookies;
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.Http;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Microsoft.EntityFrameworkCore;
using Microsoft.AspNetCore.DataProtection;
using
Microsoft.AspNetCore.DataProtection.AuthenticatedEncryption.ConfigurationMode
l;
using Microsoft.AspNetCore.DataProtection.AuthenticatedEncryption;
```

```
namespace CtoxWebApp
{
    public class Startup
    {
        public Startup(IConfiguration configuration)
        {
            var builder = new ConfigurationBuilder()
                .AddConfiguration(configuration)
                .AddJsonFile("devsettings.json", true)
                .AddEnvironmentVariables();

            Configuration = builder.Build();
        }

        private IConfiguration Configuration { get; }

        // This method gets called by the runtime. Use this method to add
services to the container.
        public void ConfigureServices(IServiceCollection services)
        {
            services
                .AddDataProtection()
                .UseCryptographicAlgorithms(new
AuthenticatedEncryptorConfiguration()
                {
                    EncryptionAlgorithm = EncryptionAlgorithm.AES_256_CBC,
                    ValidationAlgorithm = ValidationAlgorithm.HMACSHA256
                });

            services.AddResponseCompression(o => o.EnableForHttps = true);

            services.AddTransient(p => Configuration);
            services.AddTransient<IHashService, HashService>();
            services.AddTransient<IStringCompressService,
GzipCompressService>();
            services.AddTransient<ApiKey>();
            services.AddSingleton<IParseAsyncService, ParseAsyncService>();
            services.AddSingleton<IEmailSenderService, EmailSenderService>();
            services.AddSingleton<RestrictionService>();

            services

.AddAuthentication(CookieAuthenticationDefaults.AuthenticationScheme)
                .AddCookie(o =>
                {
                    o.LoginPath = new PathString("/Auth/Login");
                    o.Events.OnRedirectToAccessDenied = context =>
                    {
                        context.Response.StatusCode = 404;
                        return Task.CompletedTask;
                    };
                });

            services
                .AddControllersWithViews()
                .AddControllersAsServices()
                .AddRazorRuntimeCompilation();
```

```
            var appContextConnection =
Configuration.GetConnectionString("AppContext");
            services.AddDbContext<AppDbContext>(options =>
                options
                    .UseLazyLoadingProxies()
                    .UseMySql(appContextConnection,
ServerVersion.AutoDetect(appContextConnection)));
        }

        // This method gets called by the runtime. Use this method to
configure the HTTP request pipeline.
        public void Configure(IApplicationBuilder app, IWebHostEnvironment
env)
        {
            app.EnsureConnected<AppDbContext>();
            app.EnsureMigrated<AppDbContext>();

            app.UseResponseCompression();

            if (env.IsDevelopment())
            {
                app.UseDeveloperExceptionPage();
            }

            app.UseStatusCodePagesWithReExecute("/error", "?code={0}");

            app.UseDefaultFiles();

            app.UseStaticFiles();

            app.UseRouting();

            app.UseAuthentication();
            app.UseAuthorization();

            app.UseEndpoints(endpoints =>
            {
                endpoints.MapControllerRoute(
                    "verification",
                    "Verify/{verificationString:required}",
                    new { controller = "Auth", action = "Verify" });
                endpoints.MapControllerRoute(
                    "restorepassword",
                    "Reset/{resetString:required}",
                    new { controller = "Auth", action = "Reset" });
                endpoints.MapControllerRoute(
                    name: "default",
                    pattern: "{controller}/{action}"
                );
                endpoints.MapControllers();
            });
        }
    }
}
```

ПРИЛОЖЕНИЕ Б. ИСХОДНЫЙ КОД ПАРСЕРА

```
%e  1019
%p  2807
%n  371
%k  284
%a  1213
%o  1117

O   [0-7]
D   [0-9]
NZ  [1-9]
L   [a-zA-Z_]
A   [a-zA-Z_0-9]
H   [a-fA-F0-9]
HP  (0[xX])
E   ([Ee][+-]?{D}+)
P   ([Pp][+-]?{D}+)
FS  (f|F|l|L)
IS  (((u|U)(l|L|ll|LL)?)|((l|L|ll|LL)(u|U)?))
CP  (u|U|L)
SP  (u8|u|U|L)
ES  (\\(['"\?\\abfnrtv]|[0-7]{1,3}|x[a-fA-F0-9]+))
WS  [ \t\v\n\f]

%{
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "c_parser.tab.h"

extern void yyerror(char *);
extern char *concatn(int n, ...);
void comment(void);
void preprocessor(void);

#define COPYVALUE yylval = concatn(1, yytext)
%}

%option noyywrap
%option nounput
%option yylineno

%%

"#"             {preprocessor();}

"/*"             {comment();}
"//".*           {/* consume // comment*/}

"auto"                  { COPYVALUE; return AUTO; }
"break"                 { COPYVALUE; return BREAK; }
"case"                  { COPYVALUE; return CASE; }
"char"                  { COPYVALUE; return CHAR; }
"const"                 { COPYVALUE; return CONST; }
```

```
"continue"              { COPYVALUE; return CONTINUE; }
"default"               { COPYVALUE; return DEFAULT; }
"do"                    { COPYVALUE; return DO; }
"double"                { COPYVALUE; return DOUBLE; }
"else"                  { COPYVALUE; return ELSE; }
"enum"                  { COPYVALUE; return ENUM; }
"extern"                { COPYVALUE; return EXTERN; }
"float"                 { COPYVALUE; return FLOAT; }
"for"                   { COPYVALUE; return FOR; }
"goto"                  { COPYVALUE; return GOTO; }
"if"                    { COPYVALUE; return IF; }
"inline"                { COPYVALUE; return INLINE; }
"int"                   { COPYVALUE; return INT; }
"long"                  { COPYVALUE; return LONG; }
"register"              { COPYVALUE; return REGISTER; }
"restrict"              { COPYVALUE; return RESTRICT; }
"return"                { COPYVALUE; return RETURN; }
"short"                 { COPYVALUE; return SHORT; }
"signed"                { COPYVALUE; return SIGNED; }
"sizeof"                { COPYVALUE; return SIZEOF; }
"static"                { COPYVALUE; return STATIC; }
"struct"                { COPYVALUE; return STRUCT; }
"switch"                { COPYVALUE; return SWITCH; }
"typedef"               { COPYVALUE; return TYPEDEF; }
"union"                 { COPYVALUE; return UNION; }
"unsigned"              { COPYVALUE; return UNSIGNED; }
"void"                  { COPYVALUE; return VOID; }
"volatile"              { COPYVALUE; return VOLATILE; }
"while"                 { COPYVALUE; return WHILE; }
"_Alignas"       { COPYVALUE; return ALIGNAS; }
"_Alignof"       { COPYVALUE; return ALIGNOF; }
"_Atomic"        { COPYVALUE; return ATOMIC; }
"_Bool"                 { COPYVALUE; return BOOL; }
"_Complex"              { COPYVALUE; return COMPLEX; }
"_Generic"       { COPYVALUE; return GENERIC; }
"_Imaginary"    { COPYVALUE; return IMAGINARY; }
"_Noreturn"      { COPYVALUE; return NORETURN; }
"_Static_assert" { COPYVALUE; return STATIC_ASSERT; }
"_Thread_local" { COPYVALUE; return THREAD_LOCAL; }
"__func__"       { COPYVALUE; return FUNC_NAME; }

{L}{A}*                                         { COPYVALUE; return
IDENTIFIER; }
{HP}{H}+{IS}?                                   { COPYVALUE; return I_CONSTANT;
}
{NZ}{D}*{IS}?                                   { COPYVALUE; return I_CONSTANT;
}
"0"{O}*{IS}?                                    { COPYVALUE; return I_CONSTANT;
}
{CP}?"'"([^'\\\n]|{ES})+"'"                 { COPYVALUE; return I_CONSTANT; }
{D}+{E}{FS}?                                    { COPYVALUE; return F_CONSTANT;
}
{D}*"."{D}+{E}?{FS}?                        { COPYVALUE; return F_CONSTANT; }
{D}+"."{E}?{FS}?                                { COPYVALUE; return F_CONSTANT;
}
{HP}{H}+{P}{FS}?                                { COPYVALUE; return F_CONSTANT;
}
```

```
{HP}{H}*"."{H}+{P}{FS}?                          { COPYVALUE; return
F_CONSTANT; }
{HP}{H}+"."{P}{FS}?                            { COPYVALUE; return F_CONSTANT;
}
({SP}?\"([^"\\\n]|{ES})*\"{WS}*)+      { COPYVALUE; return STRING_LITERAL; }

"..."                   { COPYVALUE; return ELLIPSIS; }
">>="                   { yylval = concatn(1, "&gt;&gt;="); return
RIGHT_ASSIGN; }
"<<="                   { yylval = concatn(1, "&lt;&lt;="); return LEFT_ASSIGN;
}
"+="                    { COPYVALUE; return ADD_ASSIGN; }
"-="                    { COPYVALUE; return SUB_ASSIGN; }
"*="                    { COPYVALUE; return MUL_ASSIGN; }
"/="                    { COPYVALUE; return DIV_ASSIGN; }
"%="                    { COPYVALUE; return MOD_ASSIGN; }
"&="                    { yylval = concatn(1, "&amp;=");; return AND_ASSIGN; }
"^="                    { COPYVALUE; return XOR_ASSIGN; }
"|="                    { COPYVALUE; return OR_ASSIGN; }
">>"                    { yylval = concatn(1, "&gt;&gt;"); return RIGHT_OP; }
"<<"                    { yylval = concatn(1, "&lt;&lt;"); return LEFT_OP; }
"++"                    { COPYVALUE; return INC_OP; }
"--"                    { COPYVALUE; return DEC_OP; }
"->"                    { yylval = concatn(1, "-&gt;"); return PTR_OP; }
"&&"                    { yylval = concatn(1, "&amp;&amp;"); return AND_OP; }
"||"                    { COPYVALUE; return OR_OP; }
"<="                    { yylval = concatn(1, "&lt;="); return LE_OP; }
">="                    { yylval = concatn(1, "&gt;="); return GE_OP; }
"=="                    { COPYVALUE; return EQ_OP; }
"!="                    { COPYVALUE; return NE_OP; }
";"                       { COPYVALUE; return ';'; }
"{"                     { COPYVALUE; return '{'; }
"}"                     { COPYVALUE; return '}'; }
","                       { COPYVALUE; return ','; }
":"                       { COPYVALUE; return ':'; }
"="                       { COPYVALUE; return '='; }
"("                       { COPYVALUE; return '('; }
")"                       { COPYVALUE; return ')'; }
"["                     { COPYVALUE; return '['; }
"]"                     { COPYVALUE; return ']'; }
"."                       { COPYVALUE; return '.'; }
"&"                       { yylval = concatn(1, "&amp;"); return '&'; }
"!"                       { COPYVALUE; return '!'; }
"~"                       { COPYVALUE; return '~'; }
"-"                       { COPYVALUE; return '-'; }
"+"                     { COPYVALUE; return '+'; }
"*"                     { COPYVALUE; return '*'; }
"/"                     { COPYVALUE; return '/'; }
"%"                     { COPYVALUE; return '%'; }
"<"                     { yylval = concatn(1, "&lt;"); return '<'; }
">"                     { yylval = concatn(1, "&gt;"); return '>'; }
"^"                     { COPYVALUE; return '^'; }
"|"                       { COPYVALUE; return '|'; }
"?"                       { COPYVALUE; return '?'; }

{WS}+               { /* ignore white spaces */  }
.                         { /* discard bad chars */ }
```

```
%%

void comment(void) {
    int c;
    while ((c = input()) != EOF) {
        if (c == '*') {
            while ((c = input()) == '*')
                ;

            if (c == '/')
                return;

            if (c == EOF)
                break;
        }
    }
    yyerror("unterminated comment");
}

void preprocessor(void) {
    int c;
    while((c = input()) != EOF) {
        if(c == '\\') {
            c = input();
            c = input();
        }
        if(c == '\n' || c == '\r')
            return;
    }
}

%define api.value.type {char *}

%token  IDENTIFIER I_CONSTANT F_CONSTANT STRING_LITERAL FUNC_NAME SIZEOF
%token  PTR_OP INC_OP DEC_OP LEFT_OP RIGHT_OP LE_OP GE_OP EQ_OP NE_OP
%token  AND_OP OR_OP MUL_ASSIGN DIV_ASSIGN MOD_ASSIGN ADD_ASSIGN
%token  SUB_ASSIGN LEFT_ASSIGN RIGHT_ASSIGN AND_ASSIGN
%token  XOR_ASSIGN OR_ASSIGN

%token  TYPEDEF EXTERN STATIC AUTO REGISTER INLINE
%token  CONST RESTRICT VOLATILE
%token  BOOL CHAR SHORT INT LONG SIGNED UNSIGNED FLOAT DOUBLE VOID
%token  COMPLEX IMAGINARY
%token  STRUCT UNION ENUM ELLIPSIS

%token CASE DEFAULT IF ELSE SWITCH WHILE DO FOR GOTO CONTINUE BREAK RETURN

%token  ALIGNAS ALIGNOF ATOMIC GENERIC NORETURN STATIC_ASSERT THREAD_LOCAL

%expect 2

%start program_unit

%{
#include <stdio.h>
#include <stdlib.h>
```

```
#include <string.h>
#include <stdarg.h>

extern int yylineno;

int yylex();
void yyerror(char *);

char *concatn(int n, ...);
void freen(int n, ...);
char *createfunc(char *, char *, char *);
void addfunccall(char *, char *);
char *createexp(char *);
char *createdecl(char *);
char *createifelse(char *, char *, char *);
char *createswitch(char *, char *);
char *createcycle(int, char *, char *);

char *currentcalls = NULL;

#define EXPS "<expression"
#define EXPE "</expression>"
#define HASC "hascalls = \"true\">"
#define DHAC "hascalls = \"false\">"
#define TEXS "<text>"
#define TEXE "</text>"

#define DECS "<declaration"
#define DECE "</declaration>"

#define FUNS "<function>"
#define FUNE "</function>"
#define PROS "<prototype>"
#define PROE "</prototype>"
#define BODS "<body>"
#define BODE "</body>"
#define NBOD "<body/>"

#define CALS "<funccall>"
#define CALE "</funccall>"
#define NAMS "<name>"
#define NAME "</name>"
#define ARGS "<arguments>"
#define ARGE "</arguments>"
#define NARG "<arguments/>"

#define IFSS "<if>"
#define IFSE "</if>"
#define CONS "<condition>"
#define CONE "</condition>"
#define IFBS "<ifbody>"
#define IFBE "</ifbody>"
#define NIFB "<ifbody/>"
#define ELBS "<elsebody>"
#define ELBE "</elsebody>"
#define NELB "<elsebody/>"
```

```
#define SWIS "<switch>"
#define SWIE "</switch>"
#define LABS "<labelstatement>"
#define LABE "</labelstatement>"

#define GOTS "<goto>"
#define GOTE "</gote>"
#define CONT "<continue/>"
#define BREA "<break/>"
#define RETS "<return>"
#define RETE "</return>"
#define NRET "<return/>"

#define CYCS "<cycle"
#define PREC "pre = \"true\">"
#define POSC "pre = \"false\">"
#define CYCE "</cycle>"
%}

%locations

%%

primary_expression
        : IDENTIFIER                    {$$ = concatn(1, $1); free($1);}

        | constant                              {$$ = concatn(1, $1); free($1);}
        | string                                {$$ = concatn(1, $1); free($1);}
        | '(' expression ')'    {$$ = concatn(3, $1, $2, $3); freen(3, $1, $2,
$3);}
        | generic_selection             {$$ = concatn(1, $1); free($1);}
        ;

constant
        : I_CONSTANT            { $$ = concatn(1, $1); free($1);}
        | F_CONSTANT            { $$ = concatn(1, $1); free($1);}
        ;

string
        : STRING_LITERAL        { $$ = concatn(1, $1); free($1);}
        | FUNC_NAME                     { $$ = concatn(1, $1); free($1);}
        ;

generic_selection
        : GENERIC '(' assignment_expression ',' generic_assoc_list ')'
        {$$ = concatn(6, $1, $2, $3, $4, $5, $6); freen(6, $1, $2, $3, $4, $5,
$6);}
        ;

generic_assoc_list
        : generic_association
        {$$ = concatn(1, $1); free($1);}
        | generic_assoc_list ',' generic_association  {$$ = concatn(3, $1, $2,
$3); freen(3, $1, $2, $3);}
        ;

generic_association
```

```
        : type_name ':' assignment_expression                   {$$ = concatn(3,
$1, $2, $3); freen(3, $1, $2, $3);}
        | DEFAULT ':' assignment_expression                             {$$ =
concatn(3, $1, $2, $3); freen(3, $1, $2, $3);}
        ;

postfix_expression
        : primary_expression                                    {$$ =
concatn(1, $1); free($1);}
        | postfix_expression '[' expression ']'
        {$$ = concatn(4, $1, $2, $3, $4); freen(4, $1, $2, $3, $4);}
        | postfix_expression '(' ')'
        {$$ = concatn(3, $1, $2, $3); addfunccall($1, NULL); freen(3, $1, $2,
$3);}
        | postfix_expression '(' argument_expression_list ')' {$$ = concatn(4,
$1, $2, $3, $4); addfunccall($1, $3); freen(4, $1, $2, $3, $4);}
        | postfix_expression '.' IDENTIFIER
        {$$ = concatn(3, $1, $2, $3); freen(3, $1, $2, $3);}
        | postfix_expression PTR_OP IDENTIFIER
        {$$ = concatn(3, $1, $2, $3); freen(3, $1, $2, $3);}
        | postfix_expression INC_OP
                {$$ = concatn(2, $1, $2); freen(2, $1, $2);}
        | postfix_expression DEC_OP
                {$$ = concatn(2, $1, $2); freen(2, $1, $2);}
        | '(' type_name ')' '{' initializer_list '}'                    {$$ =
concatn(6, $1, $2, $3, $4, $5, $6); freen(6, $1, $2, $3, $4, $5, $6);}
        | '(' type_name ')' '{' initializer_list ',' '}'                {$$ =
concatn(7, $1, $2, $3, $4, $5, $6, $7); freen(7, $1, $2, $3, $4, $5, $6,
$7);}
        ;

argument_expression_list
        : assignment_expression
                        {$$ = concatn(1, $1); free($1);}
        | argument_expression_list ',' assignment_expression  {$$ = concatn(3,
$1, $2, $3); freen(3, $1, $2, $3);}
        ;

unary_expression
        : postfix_expression                                    {$$ = concatn(1, $1);
free($1);}
        | INC_OP unary_expression                               {$$ = concatn(2, $1,
$2); freen(2, $1, $2);}
        | DEC_OP unary_expression                               {$$ = concatn(2, $1,
$2); freen(2, $1, $2);}
        | unary_operator cast_expression        {$$ = concatn(2, $1, $2);
freen(2, $1, $2);}
        | SIZEOF unary_expression                               {$$ = concatn(2, $1,
$2); freen(2, $1, $2);}
        | SIZEOF '(' type_name ')'                              {$$ = concatn(4, $1, $2,
$3, $4); freen(4, $1, $2, $3, $4);}
        | ALIGNOF '(' type_name ')'                             {$$ = concatn(4, $1, $2,
$3, $4); freen(4, $1, $2, $3, $4);}
        ;

unary_operator
        : '&'   { $$ = concatn(1, $1); free($1);}
```

```
        | '*'   { $$ = concatn(1, $1); free($1);}
        | '+'   { $$ = concatn(1, $1); free($1);}
        | '-'   { $$ = concatn(1, $1); free($1);}
        | '~'   { $$ = concatn(1, $1); free($1);}
        | '!'   { $$ = concatn(1, $1); free($1);}
        ;

cast_expression
        : unary_expression                                              {$$ =
concatn(1, $1); free($1);}
        | '(' type_name ')' cast_expression             {$$ = concatn(4, $1, $2,
$3, $4); freen(4, $1, $2, $3, $4);}
        ;

multiplicative_expression
        : cast_expression
                {$$ = concatn(1, $1); free($1);}
        | multiplicative_expression '*' cast_expression                 {$$ =
concatn(3, $1, $2, $3); freen(3, $1, $2, $3);}
        | multiplicative_expression '/' cast_expression                 {$$ =
concatn(3, $1, $2, $3); freen(3, $1, $2, $3);}
        | multiplicative_expression '%' cast_expression                 {$$ =
concatn(3, $1, $2, $3); freen(3, $1, $2, $3);}
        ;

additive_expression
        : multiplicative_expression
                {$$ = concatn(1, $1); free($1);}
        | additive_expression '+' multiplicative_expression             {$$ =
concatn(3, $1, $2, $3); freen(3, $1, $2, $3);}
        | additive_expression '-' multiplicative_expression             {$$ =
concatn(3, $1, $2, $3); freen(3, $1, $2, $3);}
        ;

shift_expression
        : additive_expression
        { $$ = concatn(1, $1); free($1);}
        | shift_expression LEFT_OP additive_expression                  {$$ =
concatn(3, $1, $2, $3); freen(3, $1, $2, $3);}
        | shift_expression RIGHT_OP additive_expression                 {$$ =
concatn(3, $1, $2, $3); freen(3, $1, $2, $3);}
        ;

relational_expression
        : shift_expression
        { $$ = concatn(1, $1); free($1);}
        | relational_expression '<' shift_expression  {$$ = concatn(3, $1, $2,
$3); freen(3, $1, $2, $3);}
        | relational_expression '>' shift_expression  {$$ = concatn(3, $1, $2,
$3); freen(3, $1, $2, $3);}
        | relational_expression LE_OP shift_expression          {$$ = concatn(3,
$1, $2, $3); freen(3, $1, $2, $3);}
        | relational_expression GE_OP shift_expression          {$$ = concatn(3,
$1, $2, $3); freen(3, $1, $2, $3);}
        ;

equality_expression
```

```
        : relational_expression
                { $$ = concatn(1, $1); free($1);}
        | equality_expression EQ_OP relational_expression     {$$ = concatn(3,
$1, $2, $3); freen(3, $1, $2, $3);}
        | equality_expression NE_OP relational_expression     {$$ = concatn(3,
$1, $2, $3); freen(3, $1, $2, $3);}
        ;

and_expression
        : equality_expression                                          { $$ =
concatn(1, $1); free($1);}
        | and_expression '&' equality_expression      {$$ = concatn(3, $1, $2,
$3); freen(3, $1, $2, $3);}
        ;

exclusive_or_expression
        : and_expression
        { $$ = concatn(1, $1); free($1);}
        | exclusive_or_expression '^' and_expression  {$$ = concatn(3, $1, $2,
$3); freen(3, $1, $2, $3);}
        ;

inclusive_or_expression
        : exclusive_or_expression
                { $$ = concatn(1, $1); free($1);}
        | inclusive_or_expression '|' exclusive_or_expression {$$ = concatn(3,
$1, $2, $3); freen(3, $1, $2, $3);}
        ;

logical_and_expression
        : inclusive_or_expression
                        { $$ = concatn(1, $1); free($1);}
        | logical_and_expression AND_OP inclusive_or_expression
        {$$ = concatn(3, $1, $2, $3); freen(3, $1, $2, $3);}
        ;

logical_or_expression
        : logical_and_expression
                { $$ = concatn(1, $1); free($1);}
        | logical_or_expression OR_OP logical_and_expression  {$$ = concatn(3,
$1, $2, $3); freen(3, $1, $2, $3);}
        ;

conditional_expression
        : logical_or_expression
                                                { $$ = concatn(1, $1);
free($1);}
        | logical_or_expression '?' expression ':' conditional_expression
        {$$ = concatn(5, $1, $2, $3, $4, $5); freen(5, $1, $2, $3, $4, $5);}
        ;

assignment_expression
        : conditional_expression
                                { $$ = concatn(1, $1); free($1);}
        | unary_expression assignment_operator assignment_expression {$$ =
concatn(3, $1, $2, $3); freen(3, $1, $2, $3);}
        ;
```

```
assignment_operator
        : '='                   { $$ = concatn(1, $1); free($1);}
        | MUL_ASSIGN    { $$ = concatn(1, $1); free($1);}
        | DIV_ASSIGN    { $$ = concatn(1, $1); free($1);}
        | MOD_ASSIGN    { $$ = concatn(1, $1); free($1);}
        | ADD_ASSIGN    { $$ = concatn(1, $1); free($1);}
        | SUB_ASSIGN    { $$ = concatn(1, $1); free($1);}
        | LEFT_ASSIGN   { $$ = concatn(1, $1); free($1);}
        | RIGHT_ASSIGN  { $$ = concatn(1, $1); free($1);}
        | AND_ASSIGN    { $$ = concatn(1, $1); free($1);}
        | XOR_ASSIGN    { $$ = concatn(1, $1); free($1);}
        | OR_ASSIGN             { $$ = concatn(1, $1); free($1);}
        ;

expression
        : assignment_expression                                         { $$ =
concatn(1, $1); free($1);}
        | expression ',' assignment_expression{$$ = concatn(3, $1, $2, $3);
freen(3, $1, $2, $3);}
        ;

constant_expression
        : conditional_expression        { $$ = concatn(1, $1); free($1);}
        ;

declaration
        : declaration_specifiers ';'
        {$$ = concatn(1, $1); freen(2, $1, $2);}
        | declaration_specifiers init_declarator_list ';'       {$$ = concatn(2,
$1, $2); freen(3, $1, $2, $3);}
        | static_assert_declaration
        { $$ = concatn(1, $1); free($1);}
        ;

declaration_specifiers
        : storage_class_specifier declaration_specifiers        {$$ = concatn(2,
$1, $2); freen(2, $1, $2);}
        | storage_class_specifier
        { $$ = concatn(1, $1); free($1);}
        | type_specifier declaration_specifiers
        {$$ = concatn(2, $1, $2); freen(2, $1, $2);}
        | type_specifier
                { $$ = concatn(1, $1); free($1);}
        | type_qualifier declaration_specifiers
        {$$ = concatn(2, $1, $2); freen(2, $1, $2);}
        | type_qualifier
                { $$ = concatn(1, $1); free($1);}
        | function_specifier declaration_specifiers                     {$$ =
concatn(2, $1, $2); freen(2, $1, $2);}
        | function_specifier
        { $$ = concatn(1, $1); free($1);}
        | alignment_specifier declaration_specifiers            {$$ = concatn(2,
$1, $2); freen(2, $1, $2);}
        | alignment_specifier
        {$$ = concatn(1, $1); free($1);}
        ;
```

```
init_declarator_list
        : init_declarator
        { $$ = concatn(1, $1); free($1);}
        | init_declarator_list ',' init_declarator            {$$ = concatn(3,
$1, $2, $3); freen(3, $1, $2, $3);}
        ;

init_declarator
        : declarator '=' initializer   {$$ = concatn(3, $1, $2, $3); freen(3,
$1, $2, $3);}
        | declarator                                    { $$ = concatn(1, $1);
free($1);}
        ;

storage_class_specifier
        : TYPEDEF               { $$ = concatn(1, $1); free($1);}
        | EXTERN                { $$ = concatn(1, $1); free($1);}
        | STATIC                { $$ = concatn(1, $1); free($1);}
        | THREAD_LOCAL  { $$ = concatn(1, $1); free($1);}
        | AUTO                  { $$ = concatn(1, $1); free($1);}
        | REGISTER              { $$ = concatn(1, $1); free($1);}
        ;

type_specifier
        : VOID                                                  { $$ =
concatn(1, $1); free($1);}
        | CHAR                                                  { $$ =
concatn(1, $1); free($1);}
        | SHORT                                                 { $$ =
concatn(1, $1); free($1);}
        | INT                                                   { $$ =
concatn(1, $1); free($1);}
        | LONG                                                  { $$ =
concatn(1, $1); free($1);}
        | FLOAT                                                 { $$ =
concatn(1, $1); free($1);}
        | DOUBLE                                                { $$ =
concatn(1, $1); free($1);}
        | SIGNED                                                { $$ =
concatn(1, $1); free($1);}
        | UNSIGNED                                              { $$ =
concatn(1, $1); free($1);}
        | BOOL                                                  { $$ =
concatn(1, $1); free($1);}
        | COMPLEX                                               { $$ =
concatn(1, $1); free($1);}
        | IMAGINARY                                             { $$ =
concatn(1, $1); free($1);}
        | atomic_type_specifier                         { $$ = concatn(1, $1);
free($1);}
        | struct_or_union_specifier             { $$ = concatn(1, $1);
free($1);}
        | enum_specifier                                { $$ = concatn(1, $1);
free($1);}
        ;
```

```
struct_or_union_specifier
        : struct_or_union '{' struct_declaration_list '}'
        {$$ = concatn(4, $1, $2, $3, $4); freen(4, $1, $2, $3, $4);}
        | struct_or_union IDENTIFIER '{' struct_declaration_list '}' {$$ =
concatn(5, $1, $2, $3, $4, $5); freen(5, $1, $2, $3, $4, $5);}
        | struct_or_union IDENTIFIER
                        {$$ = concatn(2, $1, $2); freen(2, $1, $2);}
        ;

struct_or_union
        : STRUCT        { $$ = concatn(1, $1); free($1);}
        | UNION         { $$ = concatn(1, $1); free($1);}
        ;

struct_declaration_list
        : struct_declaration                                                    {
$$ = concatn(1, $1); free($1);}
        | struct_declaration_list struct_declaration  {$$ = concatn(2, $1,
$2); freen(2, $1, $2);}
        ;

struct_declaration
        : specifier_qualifier_list ';'
        {$$ = concatn(1, $1); freen(2, $1, $2);}
        | specifier_qualifier_list struct_declarator_list ';' {$$ = concatn(2,
$1, $2); freen(3, $1, $2, $3);}
        | static_assert_declaration
                { $$ = concatn(1, $1); free($1);}
        ;

specifier_qualifier_list
        : type_specifier specifier_qualifier_list             {$$ = concatn(2,
$1, $2); freen(2, $1, $2);}
        | type_specifier
        { $$ = concatn(1, $1); free($1);}
        | type_qualifier specifier_qualifier_list             {$$ = concatn(2,
$1, $2); freen(2, $1, $2);}
        | type_qualifier
        { $$ = concatn(1, $1); free($1);}
        ;

struct_declarator_list
        : struct_declarator
        { $$ = concatn(1, $1); free($1);}
        | struct_declarator_list ',' struct_declarator        {$$ = concatn(3,
$1, $2, $3); freen(3, $1, $2, $3);}
        ;

struct_declarator
        : ':' constant_expression                             {$$ = concatn(2,
$1, $2); freen(2, $1, $2);}
        | declarator ':' constant_expression  {$$ = concatn(3, $1, $2, $3);
freen(3, $1, $2, $3);}
        | declarator                                                    { $$ =
concatn(1, $1); free($1);}
        ;
```

```
enum_specifier
        : ENUM '{' enumerator_list '}'                                  {$$ =
concatn(4, $1, $2, $3, $4); freen(4, $1, $2, $3, $4);}
        | ENUM '{' enumerator_list ',' '}'                              {$$ =
concatn(5, $1, $2, $3, $4, $5); freen(5, $1, $2, $3, $4, $5);}
        | ENUM IDENTIFIER '{' enumerator_list '}'                  {$$ = concatn(5,
$1, $2, $3, $4, $5); freen(5, $1, $2, $3, $4, $5);}
        | ENUM IDENTIFIER '{' enumerator_list ',' '}' {$$ = concatn(6, $1, $2,
$3, $4, $5, $6); freen(6, $1, $2, $3, $4, $5, $6);}
        | ENUM IDENTIFIER
        {$$ = concatn(2, $1, $2); freen(2, $1, $2);}
        ;

enumerator_list
        : enumerator                                                { $$ =
concatn(1, $1); free($1);}
        | enumerator_list ',' enumerator       {$$ = concatn(3, $1, $2, $3);
freen(3, $1, $2, $3);}
        ;

enumerator
        : IDENTIFIER '=' constant_expression  {$$ = concatn(3, $1, $2, $3);
freen(3, $1, $2, $3);}
        | IDENTIFIER                                                    { $$ =
concatn(1, $1); free($1);}
        ;

atomic_type_specifier
        : ATOMIC '(' type_name ')'                                  {$$ = concatn(4,
$1, $2, $3, $4); freen(4, $1, $2, $3, $4);}
        ;

type_qualifier
        : CONST                 { $$ = concatn(1, $1); free($1);}
        | RESTRICT              { $$ = concatn(1, $1); free($1);}
        | VOLATILE              { $$ = concatn(1, $1); free($1);}
        | ATOMIC                { $$ = concatn(1, $1); free($1);}
        ;

function_specifier
        : INLINE                { $$ = concatn(1, $1); free($1);}
        | NORETURN              { $$ = concatn(1, $1); free($1);}
        ;

alignment_specifier
        : ALIGNAS '(' type_name ')'                                 {$$ = concatn(4,
$1, $2, $3, $4); freen(4, $1, $2, $3, $4);}
        | ALIGNAS '(' constant_expression ')' {$$ = concatn(4, $1, $2, $3,
$4); freen(4, $1, $2, $3, $4);}
        ;

declarator
        : pointer direct_declarator           {$$ = concatn(2, $1, $2);
freen(2, $1, $2);}
        | direct_declarator                          { $$ = concatn(1, $1);
free($1);}
        ;
```

```
direct_declarator
        : IDENTIFIER
                                                                { $$ =
concatn(1, $1); free($1);}
        | '(' declarator ')'
                                                                {$$ = concatn(3,
$1, $2, $3); freen(3, $1, $2, $3);}
        | direct_declarator '[' ']'
                                                                {$$ = concatn(3,
$1, $2, $3); freen(3, $1, $2, $3);}
        | direct_declarator '[' '*' ']'
                                                                {$$ = concatn(4,
$1, $2, $3, $4); freen(4, $1, $2, $3, $4);}
        | direct_declarator '[' STATIC type_qualifier_list
assignment_expression ']'       {$$ = concatn(6, $1, $2, $3, $4, $5, $6);
freen(6, $1, $2, $3, $4, $5, $6);}
        | direct_declarator '[' STATIC assignment_expression ']'
                                {$$ = concatn(5, $1, $2, $3, $4, $5); freen(5,
$1, $2, $3, $4, $5);}
        | direct_declarator '[' type_qualifier_list '*' ']'
                                        {$$ = concatn(5, $1, $2, $3, $4, $5);
freen(5, $1, $2, $3, $4, $5);}
        | direct_declarator '[' type_qualifier_list STATIC
assignment_expression ']'       {$$ = concatn(6, $1, $2, $3, $4, $5, $6);
freen(6, $1, $2, $3, $4, $5, $6);}
        | direct_declarator '[' type_qualifier_list assignment_expression ']'
                        {$$ = concatn(5, $1, $2, $3, $4, $5); freen(5, $1, $2,
$3, $4, $5);}
        | direct_declarator '[' type_qualifier_list ']'
                                                {$$ = concatn(4, $1, $2, $3,
$4); freen(4, $1, $2, $3, $4);}
        | direct_declarator '[' assignment_expression ']'
                                        {$$ = concatn(4, $1, $2, $3, $4);
freen(4, $1, $2, $3, $4);}
        | direct_declarator '(' parameter_type_list ')'
                                                {$$ = concatn(4, $1, $2, $3,
$4); freen(4, $1, $2, $3, $4);}
        | direct_declarator '(' ')'
                                                                {$$ = concatn(3,
$1, $2, $3); freen(3, $1, $2, $3);}
        | direct_declarator '(' identifier_list ')'
                                                {$$ = concatn(4, $1, $2, $3,
$4); freen(4, $1, $2, $3, $4);}
        ;

pointer
        : '*' type_qualifier_list pointer    {$$ = concatn(3, $1, $2, $3);
freen(3, $1, $2, $3);}
        | '*' type_qualifier_list            {$$ = concatn(2, $1, $2); freen(2,
$1, $2);}
        | '*' pointer                        {$$ = concatn(2, $1, $2); freen(2,
$1, $2);}
        | '*'                                {$$ = concatn(1, $1); free($1);}
        ;
```

```
type_qualifier_list
        : type_qualifier                          { $$ = concatn(1, $1);
free($1);}
        | type_qualifier_list type_qualifier   {$$ = concatn(2, $1, $2);
freen(2, $1, $2);}
        ;


parameter_type_list
        : parameter_list ',' ELLIPSIS  {$$ = concatn(3, $1, $2, $3); freen(3,
$1, $2, $3);}
        | parameter_list                                { $$ = concatn(1, $1);
free($1);}
        ;

parameter_list
        : parameter_declaration                                                 {
$$ = concatn(1, $1); free($1);}
        | parameter_list ',' parameter_declaration    {$$ = concatn(3, $1, $2,
$3); freen(3, $1, $2, $3);}
        ;

parameter_declaration
        : declaration_specifiers declarator                             {$$ =
concatn(2, $1, $2); freen(2, $1, $2);}
        | declaration_specifiers abstract_declarator  {$$ = concatn(2, $1,
$2); freen(2, $1, $2);}
        | declaration_specifiers                                                {
$$ = concatn(1, $1); free($1);}
        ;

identifier_list
        : IDENTIFIER                                            { $$ =
concatn(1, $1); free($1);}
        | identifier_list ',' IDENTIFIER      {$$ = concatn(3, $1, $2, $3);
freen(3, $1, $2, $3);}
        ;

type_name
        : specifier_qualifier_list abstract_declarator        {$$ = concatn(2,
$1, $2); freen(2, $1, $2);}
        | specifier_qualifier_list                                              {
$$ = concatn(1, $1); free($1);}
        ;

abstract_declarator
        : pointer direct_abstract_declarator  {$$ = concatn(2, $1, $2);
freen(2, $1, $2);}
        | pointer                                                               {
$$ = concatn(1, $1); free($1);}
        | direct_abstract_declarator                  { $$ = concatn(1, $1);
free($1);}
        ;

direct_abstract_declarator
```

```
        : '(' abstract_declarator ')'

        {$$ = concatn(3, $1, $2, $3); freen(3, $1, $2, $3);}
        | '[' ']'

                        {$$ = concatn(2, $1, $2); freen(2, $1, $2);}
        | '[' '*' ']'

                {$$ = concatn(3, $1, $2, $3); freen(3, $1, $2, $3);}
        | '[' STATIC type_qualifier_list assignment_expression ']'
                                                {$$ = concatn(5, $1, $2,
$3, $4, $5); freen(5, $1, $2, $3, $4, $5);}
        | '[' STATIC assignment_expression ']'
                                                        {$$ =
concatn(4, $1, $2, $3, $4); freen(4, $1, $2, $3, $4);}
        | '[' type_qualifier_list STATIC assignment_expression ']'
                                                {$$ = concatn(5, $1, $2,
$3, $4, $5); freen(5, $1, $2, $3, $4, $5);}
        | '[' type_qualifier_list assignment_expression ']'
                                                        {$$ = concatn(4,
$1, $2, $3, $4); freen(4, $1, $2, $3, $4);}
        | '[' type_qualifier_list ']'

        {$$ = concatn(3, $1, $2, $3); freen(3, $1, $2, $3);}
        | '[' assignment_expression ']'

        {$$ = concatn(3, $1, $2, $3); freen(3, $1, $2, $3);}
        | direct_abstract_declarator '[' ']'
                                                        {$$ =
concatn(3, $1, $2, $3); freen(3, $1, $2, $3);}
        | direct_abstract_declarator '[' '*' ']'
                                                        {$$ =
concatn(4, $1, $2, $3, $4); freen(4, $1, $2, $3, $4);}
        | direct_abstract_declarator '[' STATIC type_qualifier_list
assignment_expression ']'               {$$ = concatn(6, $1, $2, $3, $4, $5,
$6); freen(6, $1, $2, $3, $4, $5, $6);}
        | direct_abstract_declarator '[' STATIC assignment_expression ']'
                                                {$$ = concatn(5, $1, $2, $3, $4,
$5); freen(5, $1, $2, $3, $4, $5);}
        | direct_abstract_declarator '[' type_qualifier_list
assignment_expression ']'                               {$$ = concatn(5, $1, $2,
$3, $4, $5); freen(5, $1, $2, $3, $4, $5);}
        | direct_abstract_declarator '[' type_qualifier_list STATIC
assignment_expression ']'               {$$ = concatn(6, $1, $2, $3, $4, $5,
$6); freen(6, $1, $2, $3, $4, $5, $6);}
        | direct_abstract_declarator '[' type_qualifier_list ']'
                                                {$$ = concatn(4, $1, $2,
$3, $4); freen(4, $1, $2, $3, $4);}
        | direct_abstract_declarator '[' assignment_expression ']'
                                                {$$ = concatn(4, $1, $2,
$3, $4); freen(4, $1, $2, $3, $4);}
        | '(' ')'

                        {$$ = concatn(2, $1, $2); freen(2, $1, $2);}
        | '(' parameter_type_list ')'

        {$$ = concatn(3, $1, $2, $3); freen(3, $1, $2, $3);}
```

```
        | direct_abstract_declarator '(' ')'
                                                                {$$ =
concatn(3, $1, $2, $3); freen(3, $1, $2, $3);}
        | direct_abstract_declarator '(' parameter_type_list ')'
                                                        {$$ = concatn(4, $1, $2,
$3, $4); freen(4, $1, $2, $3, $4);}
        ;

initializer
        : '{' initializer_list '}'                      {$$ = concatn(3, $1, $2,
$3); freen(3, $1, $2, $3);}
        | '{' initializer_list ',' '}'          {$$ = concatn(4, $1, $2, $3,
$4); freen(4, $1, $2, $3, $4);}
        | assignment_expression                                 {$$ = concatn(1,
$1);free($1);}
        ;

initializer_list
        : designation initializer
        {$$ = concatn(2, $1, $2); freen(2, $1, $2);}
        | initializer
        {$$ = concatn(1, $1); free($1);}
        | initializer_list ',' designation initializer          {$$ = concatn(4,
$1, $2, $3, $4); freen(4, $1, $2, $3, $4);}
        | initializer_list ',' initializer                              {$$ =
concatn(3, $1, $2, $3); freen(3, $1, $2, $3);}
        ;

designation
        : designator_list '='
        {$$ = concatn(2, $1, $2); freen(2, $1, $2);}
        ;

designator_list
        : designator                                                    {$$ =
concatn(1, $1); free($1);}
        | designator_list designator                    {$$ = concatn(2, $1,
$2); freen(2, $1, $2);}
        ;

designator
        : '[' constant_expression ']'           {$$ = concatn(3, $1, $2, $3);
freen(3, $1, $2, $3);}
        | '.' IDENTIFIER                                        {$$ = concatn(2,
$1, $2); freen(2, $1, $2);}
        ;

static_assert_declaration
        : STATIC_ASSERT '(' constant_expression ',' STRING_LITERAL ')' ';' {$$
= concatn(6, $1, $2, $3, $4, $5, $6);freen(7, $1, $2, $3, $4, $5, $6, $7);}
        ;

statement
        : labeled_statement                     { $$ = concatn(1, $1);
free($1);}
        | compound_statement            { $$ = concatn(1, $1); free($1);}
        | expression_statement          {$$ = createexp($1); free($1);}
```

```
        | selection_statement       { $$ = concatn(1, $1); free($1);}
        | iteration_statement       {$$ = concatn(1, $1); free($1);}
        | jump_statement                    { $$ = concatn(1, $1);
free($1);}
        ;

labeled_statement
        : IDENTIFIER ':' statement                                      {$$ =
concatn(4, LABS, $1, LABE, $3); freen(3, $1, $2, $3);}

        | CASE constant_expression ':' statement      {$$ = concatn(4, LABS,
$2, LABE, $4); freen(4, $1, $2, $3, $4);}
        | DEFAULT ':' statement
        {$$ = concatn(4, LABS, $1, LABE, $3); freen(3, $1, $2, $3);}
        ;

compound_statement
        : '{' '}'                       {$$ = concatn(1, ""); freen(2, $1,
$2);}
        | '{' block_item_list '}'       {$$ = concatn(1, $2); freen(3, $1, $2,
$3);}
        ;

block_item_list
        : block_item                    { $$ = concatn(1, $1); free($1);}
        | block_item_list block_item    {$$ = concatn(2, $1, $2); freen(2, $1,
$2);}
        ;

block_item
        : declaration   { $$ = createdecl($1); free($1);}
        | statement     { $$ = concatn(1, $1); free($1);}
        ;

expression_statement
        : ';'                           {$$ = concatn(1, ""); free($1);}
        | expression ';'        { $$ = concatn(1, $1); freen(2, $1, $2);}
        ;

selection_statement
        : IF '(' expression ')' statement ELSE statement      {$$ =
createifelse($3, $5, $7); freen(7, $1, $2, $3, $4, $5, $6, $7);}
        | IF '(' expression ')' statement
        {$$ = createifelse($3, $5, NULL); freen(5, $1, $2, $3, $4, $5);}
        | SWITCH '(' expression ')' statement                           {$$ =
createswitch($3, $5); freen(5, $1, $2, $3, $4, $5);}
        ;

iteration_statement
        : WHILE '(' expression ')' statement
                                                {$$ = createcycle(1, $3, $5);
freen(5, $1, $2, $3, $4, $5);}
        | DO statement WHILE '(' expression ')' ';'
                                                {$$ = createcycle(0, $5, $2);
freen(7, $1, $2, $3, $4, $5, $6, $7);}
```

```
        | FOR '(' expression_statement expression_statement ')' statement
                        {char *cond = concatn(4, $3, ";", $4, ";"); $$ =
createcycle(1, cond, $6); freen(7, $1, $2, $3, $4, $5, $6, cond);}
        | FOR '(' expression_statement expression_statement expression ')'
statement       {char *cond = concatn(5, $3, ";", $4, ";", $5); $$ =
createcycle(1, cond, $7); freen(8, $1, $2, $3, $4, $5, $6, $7, cond);}
        | FOR '(' declaration expression_statement ')' statement
                                {char *cond = concatn(4, $3, ";", $4, ";"); $$
= createcycle(1, cond, $6); freen(7, $1, $2, $3, $4, $5, $6, cond);}
        | FOR '(' declaration expression_statement expression ')' statement
                        {char *cond = concatn(5, $3, ";", $4, ";", $5); $$ =
createcycle(1, cond, $7); freen(8, $1, $2, $3, $4, $5, $6, $7, cond);}
        ;

jump_statement
        : GOTO IDENTIFIER ';'           {$$ = concatn(3, GOTS, $2, GOTE);
freen(3, $1, $2, $3);}
        | CONTINUE ';'                          {$$ = concatn(1, CONT); freen(2,
$1, $2);}
        | BREAK ';'                                     {$$ = concatn(1, BREA);
freen(2, $1, $2);}
        | RETURN ';'                            {$$ = concatn(1, NRET); freen(2,
$1, $2);}
        | RETURN expression ';'                 {$$ = concatn(3, RETS, $2,
RETE); freen(3, $1, $2, $3);}
        ;

program_unit
        : translation_unit              {fprintf(stderr, "%s\n", "no error
occurred");$$ = concatn(3, "<program>", $1, "</program>");printf("%s", $$);
freen(2, $1, $$);}
        ;

translation_unit
        : external_declaration                                          {$$ =
concatn(1, $1); free($1);}
        | translation_unit external_declaration             {$$ = concatn(2,
$1, $2); freen(2, $1, $2);}
        ;

external_declaration
        : function_definition  { $$ = concatn(1, $1); free($1);}
        | declaration                   { $$ = createdecl($1); free($1);}
        ;

function_definition
        : declaration_specifiers declarator declaration_list
compound_statement
        | declaration_specifiers declarator compound_statement
                {$$ = createfunc($1, $2, $3); freen(3, $1, $2, $3);}
        ;

declaration_list
        : declaration                                                   {$$ =
createdecl($1);free($1);}
        | declaration_list declaration          {$$ = concatn(4, $1, DECS, $2,
DECE); freen(2, $1, $2);}
```

```
        ;

%%

char *createcycle(int precondition, char *condition, char *body) {
        char *toreturn;
        if(strlen(body) > 0) {
                if(precondition)
                        toreturn = concatn(9, CYCS, PREC, CONS, condition,
CONE, BODS, body, BODE, CYCE);
                else
                        toreturn = concatn(9, CYCS, POSC, CONS, condition,
CONE, BODS, body, BODE, CYCE);
        } else {
                if(precondition)
                        toreturn = concatn(7, CYCS, PREC, CONS, condition,
CONE, NBOD, CYCE);
                else
                        toreturn = concatn(7, CYCS, POSC, CONS, condition,
CONE, NBOD, CYCE);
        }
        return toreturn;
}

char *createswitch(char *condition, char *body) {
        return concatn(6, SWIS, CONS, condition, CONE, body, SWIE);
}

char *createifelse(char *condition, char *ifbody, char *elsebody) {
        char *toreturn;
        if(elsebody && strlen(elsebody) > 0) {
                if(strlen(ifbody) > 0)
                        toreturn = concatn(11, IFSS, CONS, condition, CONE,
IFBS, ifbody, IFBE, ELBS, elsebody, ELBE, IFSE);
                else
                        toreturn = concatn(9, IFSS, CONS, condition, CONE,
NIFB, ELBS, elsebody, ELBE, IFSE);
        } else {
                if(strlen(ifbody) > 0)
                        toreturn = concatn(9, IFSS, CONS, condition, CONE,
IFBS, ifbody, IFBE, NELB, IFSE);
                else
                        toreturn = concatn(7, IFSS, CONS, condition, CONE,
NIFB, NELB, IFSE);
        }
        return toreturn;
}

char *createdecl(char *decl) {
        char *toreturn = NULL;
        if(currentcalls) {
                toreturn = concatn(7, DECS, HASC, currentcalls, TEXS, decl,
TEXE, DECE);
                free(currentcalls);
                currentcalls = NULL;
        } else {
                toreturn = concatn(6, DECS, DHAC, TEXS, decl, TEXE, DECE);
```

```
        }
        return toreturn;
}

char *createexp(char *exp) {
        char *toreturn = NULL;
        if(strlen(exp) > 0) {
                if(currentcalls) {
                        toreturn = concatn(7, EXPS, HASC, currentcalls, TEXS,
exp, TEXE, EXPE);
                        free(currentcalls);
                        currentcalls = NULL;
                } else {
                        toreturn = concatn(6, EXPS, DHAC, TEXS, exp, TEXE,
EXPE);
                }
        } else {
                toreturn = concatn(1, "");
        }
        return toreturn;
}

void addfunccall(char *funcname, char *args) {
        if(currentcalls) {
                if(args)
                        currentcalls = concatn(9, currentcalls, CALS, NAMS,
funcname, NAME, ARGS,  args, ARGE, CALE);
                else
                        currentcalls = concatn(7, currentcalls, CALS, NAME,
funcname, NAME, NARG, CALE);
        } else {
                if(args)
                        currentcalls = concatn(8, CALS, NAMS, funcname, NAME,
ARGS, args, ARGE, CALE);
                else
                        currentcalls = concatn(6, CALS, NAMS, funcname, NAME,
NARG, CALE);
        }
}

char *createfunc(char *specifiers, char *prototype, char *body) {
        char *toreturn;
        if(strlen(body) > 0)
                toreturn = concatn(9, FUNS, PROS, specifiers, prototype, PROE,
BODS, body, BODE, FUNE);
        else
                toreturn = concatn(7, FUNS, PROS, specifiers, prototype, PROE,
NBOD, FUNE);
        return toreturn;
}

char *concatn(int n, ...) {
    size_t flen = 0;
    va_list ap;
    va_start(ap, n);
    int i;
    for(i = 0; i < n; i++) {
```

```
        flen += strlen(va_arg(ap, char *));
    }
    va_end(ap);
    char *dest = (char *)calloc(flen + n, sizeof(char));
        if(!dest) {
                fflush(stdout);
        fprintf(stderr, "%s\n", "an error occurred, while trying to allocate
memory");
        exit(1);
        }
    flen = 0;
    va_start(ap, n);
    for(i = 0; i < n; i++) {
        char *tmp = va_arg(ap, char *);
        strcpy(dest + flen, tmp);
        flen += strlen(tmp);
        if(i != n - 1) {
            dest[flen] = ' ';
            flen++;
        }
    }
    va_end(ap);
    return dest;
}

void freen(int n, ...) {
    va_list ap;
    va_start(ap, n);
    for(int i = 0; i < n; i++) {
        char *tmp = va_arg(ap, char *);
        free(tmp);
    }
    va_end(ap);
}

void yyerror(char *str) {
        fflush(stdout);
    fprintf(stderr, "%s on line %d\n", str, yylineno);
    exit(1);
}

int main() {
        freopen("./ctox.output.xml", "w", stdout);
    return yyparse();
}
```

ПРИЛОЖЕНИЕ В