**СОДЕРЖАНИЕ**

[Введение 5](#_Toc71102798)

[1 Анализ прототипов, литературных источников и формирование требований к проектирумому программному средству 7](#_Toc71102799)

[1.1 Анализ существующих прототипов 7](#_Toc71102800)

[1.2 Постановка задачи 11](#_Toc71102801)

[2 Анализ требований к программному средству и разработка функциональных требований 12](#_Toc71102802)

[2.1 Описание функциональности программного средства 12](#_Toc71102803)

[2.2 Спецификация функциональных требований 13](#_Toc71102804)

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:

1. аутентификация и авторизация с помощью графического интерфейса;
2. графический интерфейс взаимодействия с REST службой;
3. пользовательские роли с ограничениями пользования;
4. REST служба, включающая в себя:
   1. авторизация по ключу;
   2. интерфейс взаимодействия с парсером;
   3. поиск по проведенным операциям;
5. 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.

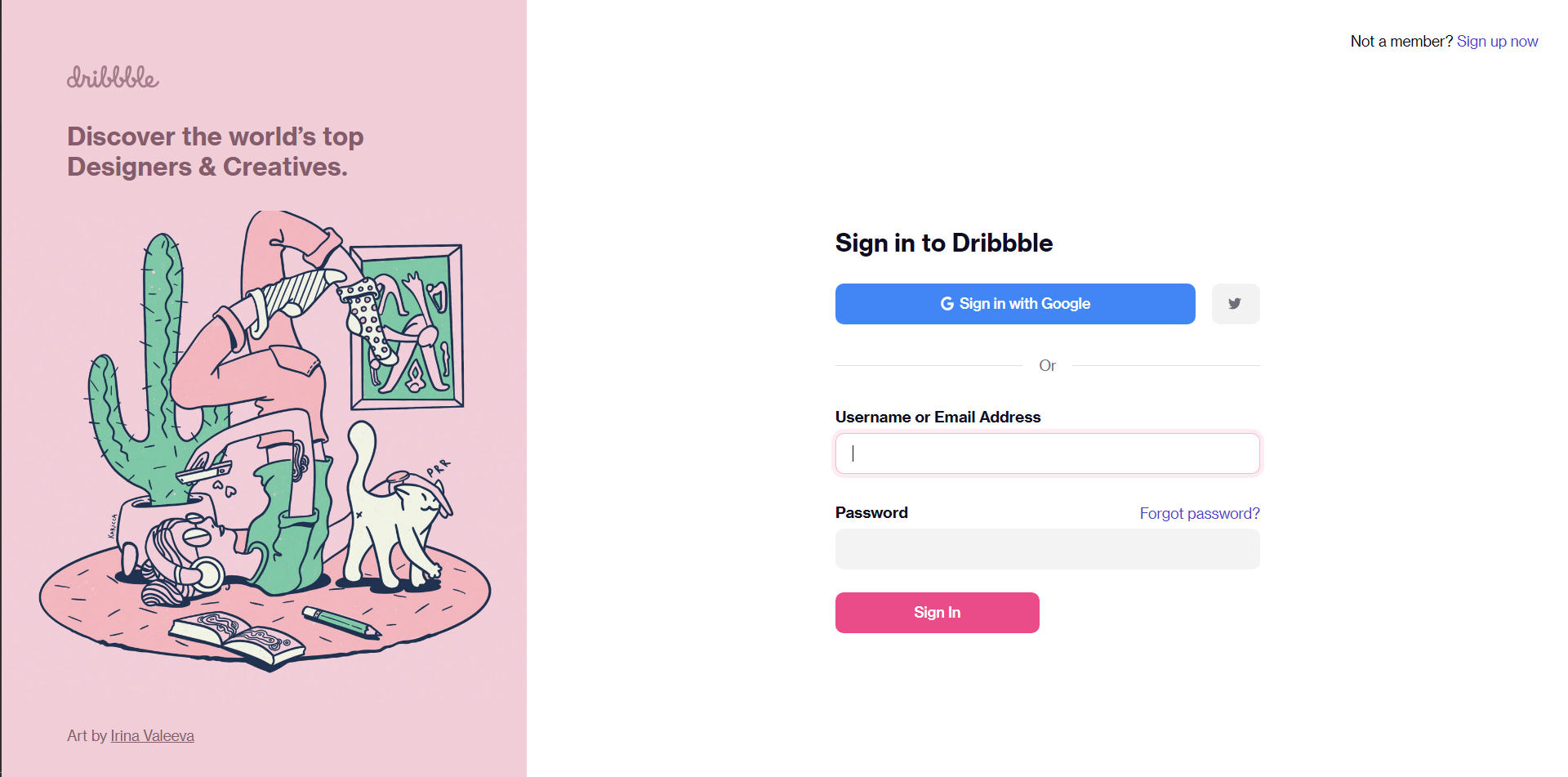


Рисунок 1.1 – Страница логина Dribble

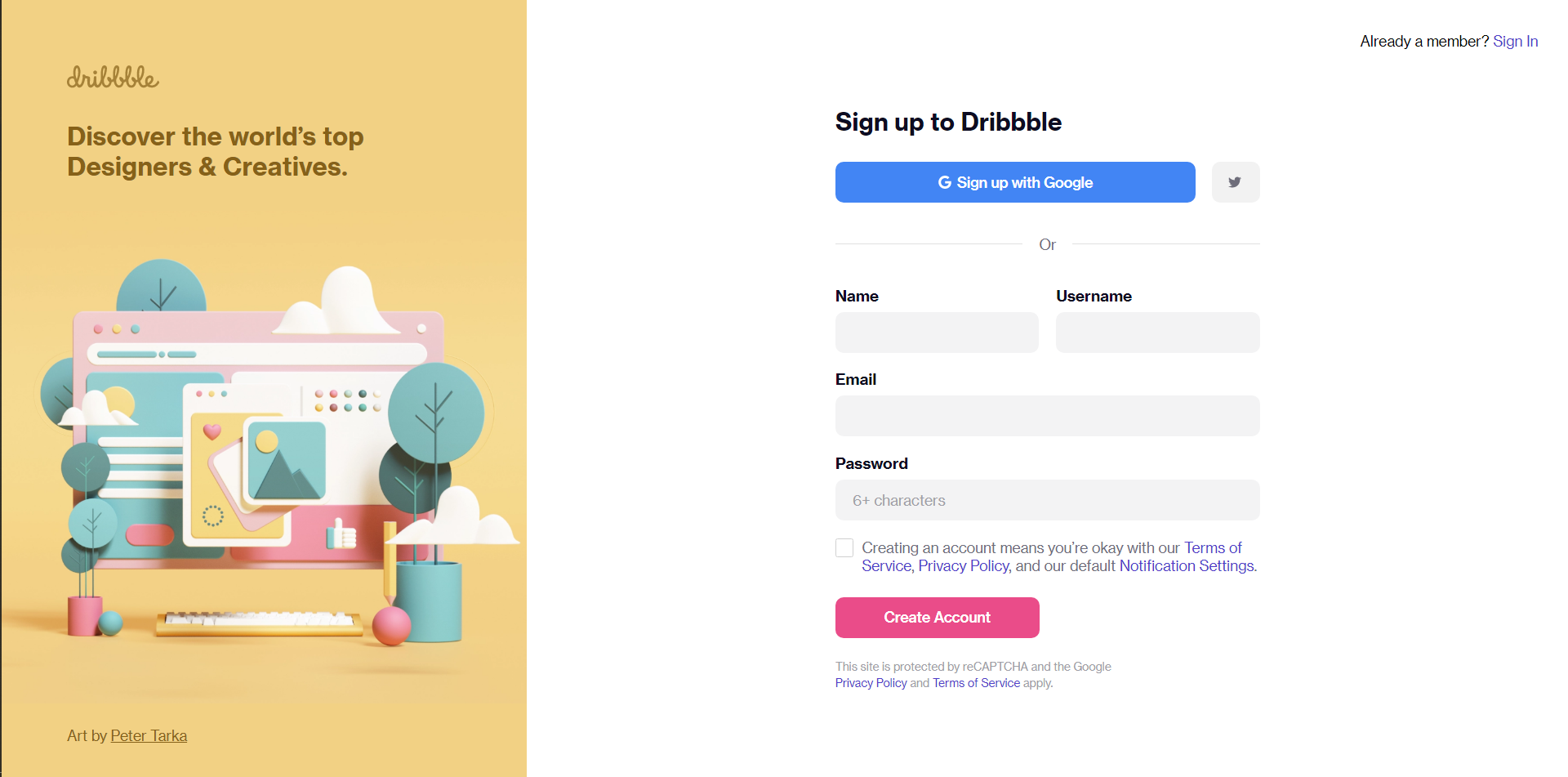


Рисунок 1.2 – Страница регистрации Dribbble

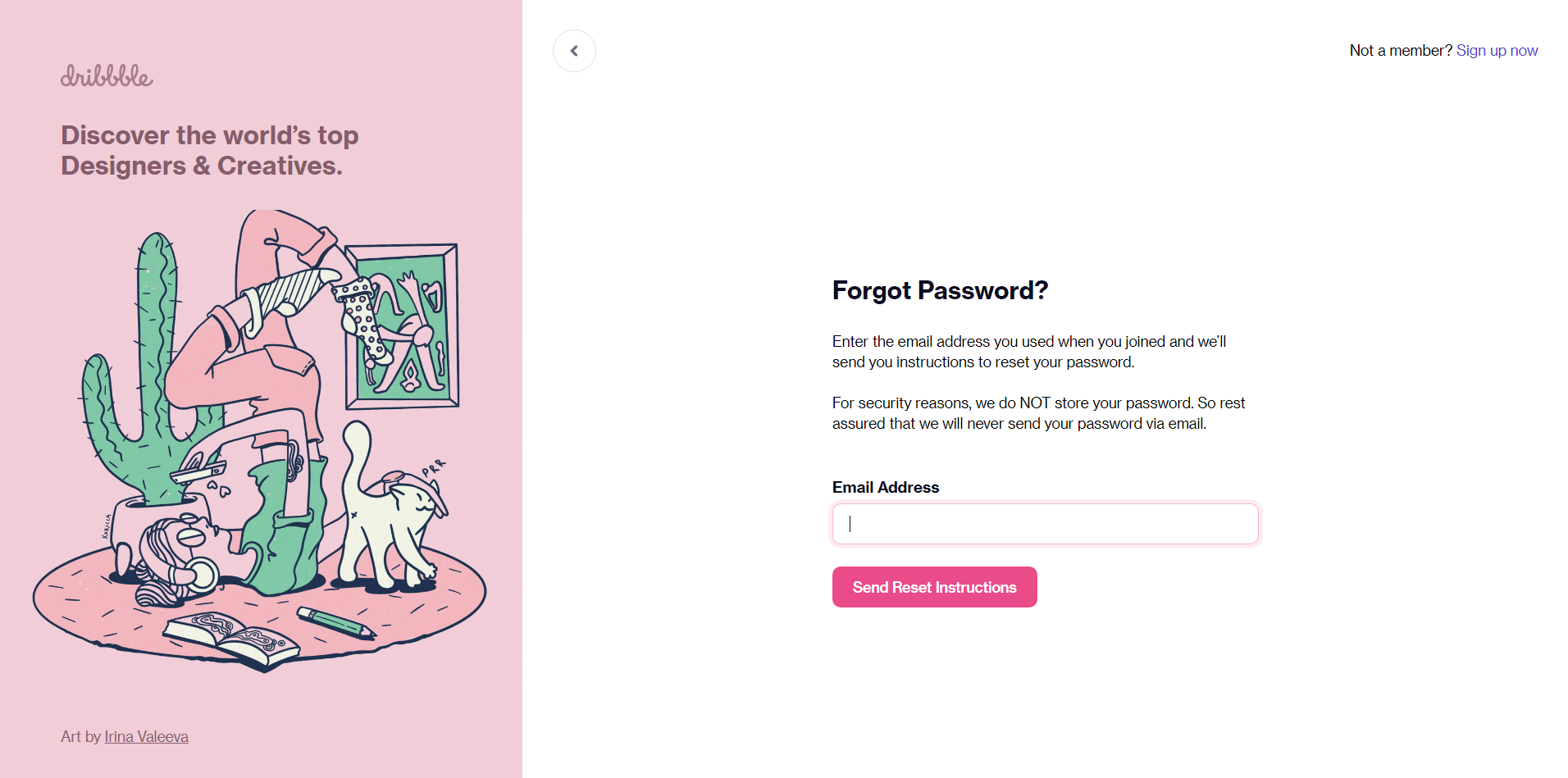


Рисунок 1.3 – Страница восстановления пароля Dribbble

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:

1. дружественный и красивый дизайн;
2. удобная форма регистрации и авторизации;
3. 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:

1. отсутствие удобного способа возврата на домашнюю страницу;
2. 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.

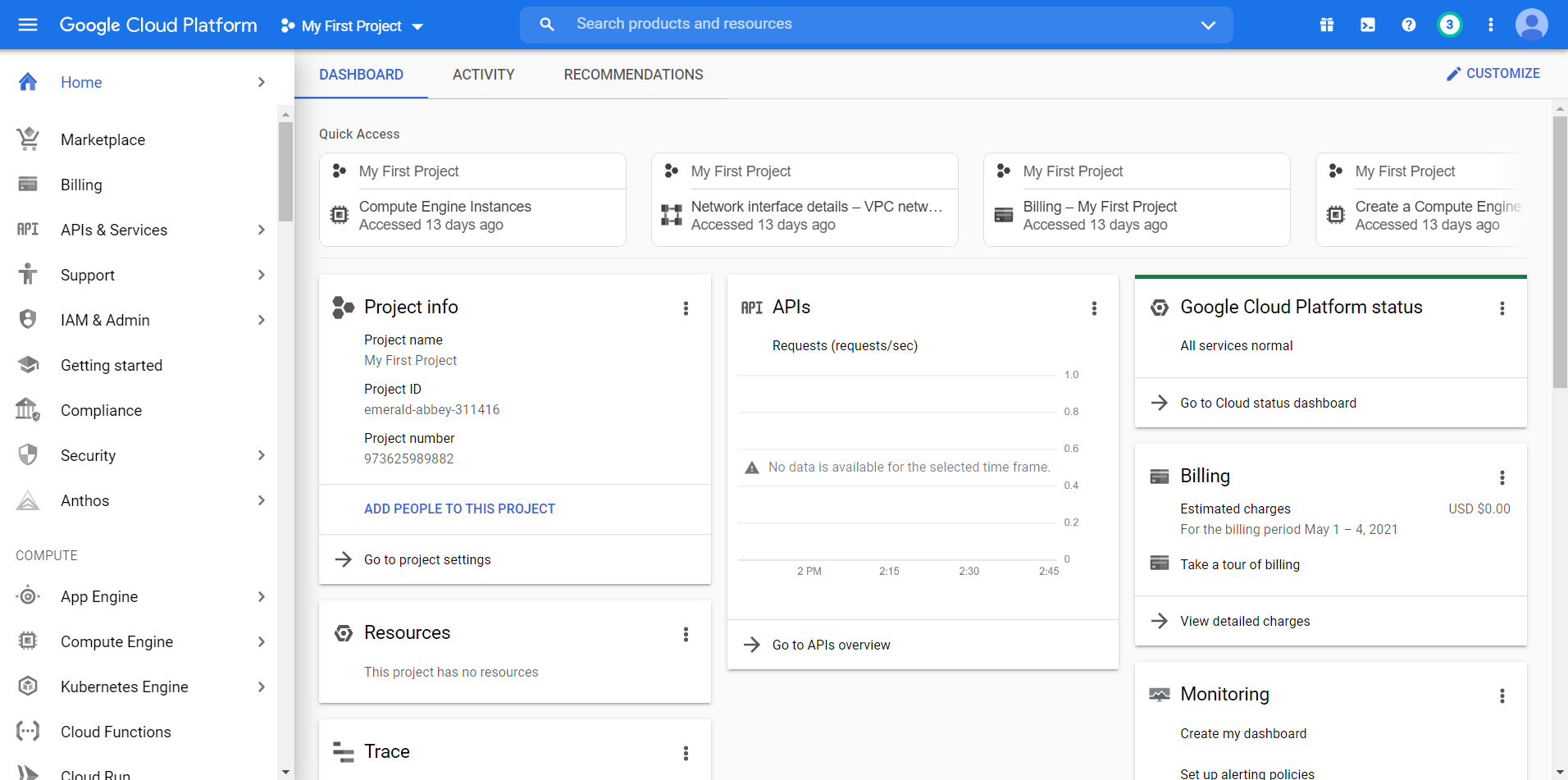


Рисунок 1.4 – Интерфейс Google Cloud

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:

1. удобный доступ к различным сервисам через меню;
2. 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.

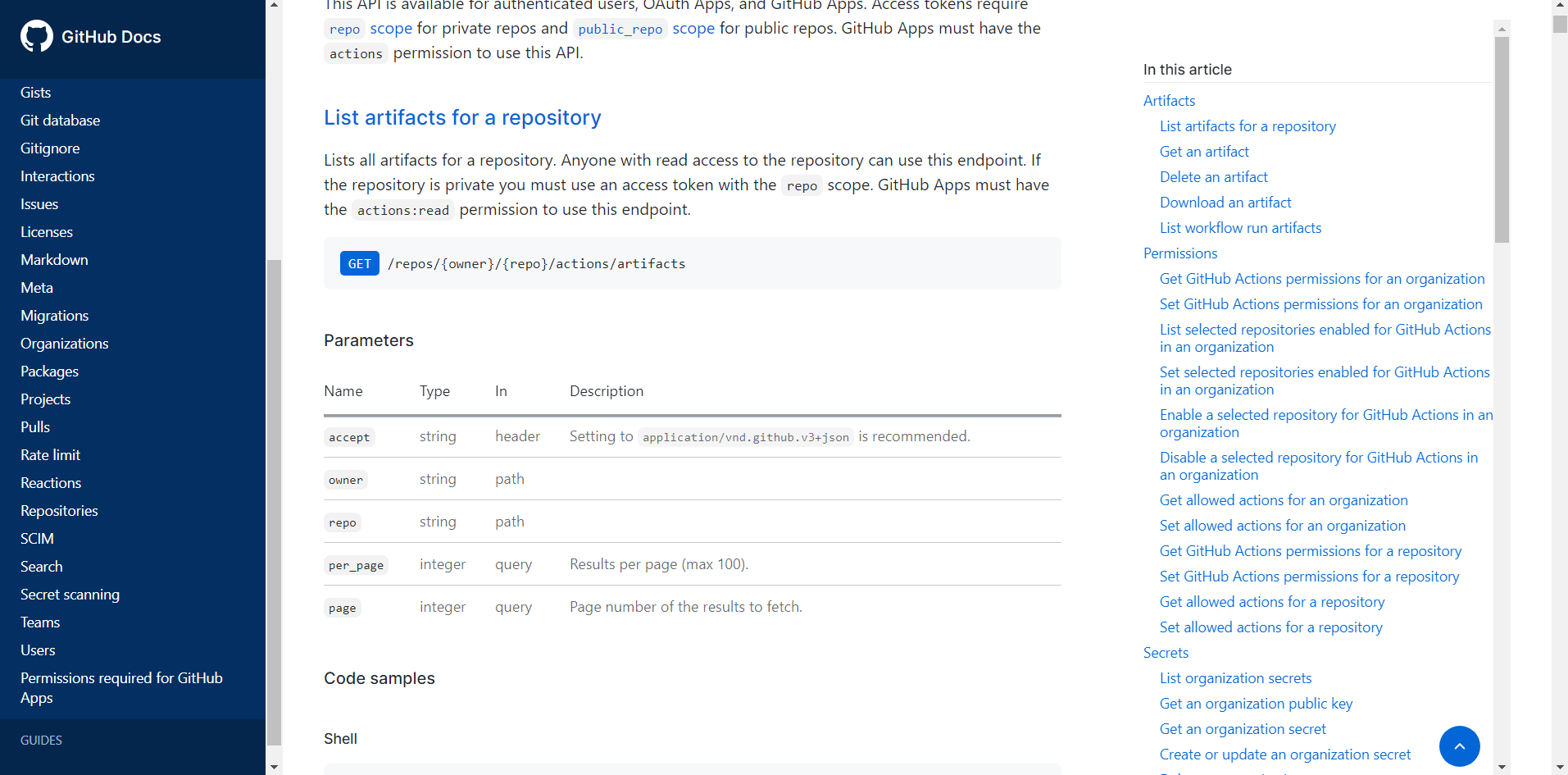


Рисунок 1.5 – Документация Github Rest

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:

1. простая аутентификация посредством ключа и одного заголовка;
2. полная документация с описанием всех точек, параметров и ошибок;
3. иерархически логичная структура точек;
4. 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:

1. 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;
2. программный интерфейс:
   1. тестовая точка доступа;
   2. авторизация по ключу;
   3. парсинг;
   4. просмотр истории запросов;
3. сервис синтаксического разбора с построением абстрактного синтаксического дерева в виде дерева XML.