МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

учреждение образования

«Гродненский государственный университет имени Янки Купалы»

Факультет математики и информатики

**Кафедра современных технологий программирования**

КОХАНОВА АННА ВИТАЛЬЕВНА

**Разработка расширения “To-Do list” для Google Chrome.**

Курсовая работа

студента 2 курса специальности

1-40 01 02 «Программное обеспечение информационных технологий»

дневной формы получения образования

|  |
| --- |
| Научный руководитель  Скращук Владислав Сергеевич, преподаватель |

Гродно, 2016

**РЕЗЮМЕ**

Тема курсовой работы

**“Разработка расширения “To-Do list” для Google Chrome.”**

Работа содержит – 15 страниц, 5 глав в основной части, 2 иллюстрации, 3 литературных источника.

Ключевые слова – To-Do List, Google Chrome расширение, пользовательский интерфейс, авторизация, аккаунт Google+, Google Drive, Google Drive API.

Цель курсовой работы – разработка расширения “To-Do list” для браузера Google Chrome с возможностью создания, редактирования, удаления и сохранения текстовых заметок на Google Drive пользователя.

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 4](#_Toc451342852)

[ГЛАВА 1. Google Chrome extension. 5](#_Toc451342853)

[1.1 Понятие и характерные черты Google Chrome расширения. 5](#_Toc451342854)

[1.2 Создание Google Chrome extension и заполнение manifest.json. 5](#_Toc451342855)

[ГЛАВА 2. Авторизация. 8](#_Toc451342856)

[2.1 Открытый OAuth протокол. 8](#_Toc451342857)

[2.2 Регистрация приложения в Google API Console. 9](#_Toc451342858)

[2.3 Авторизация через Google. 10](#_Toc451342859)

[ГЛАВА 3. Работа с Google Drive API. 11](#_Toc451342860)

[3.1 Описание Google Drive API. 11](#_Toc451342861)

[3.2 Работа с Google Drive. 11](#_Toc451342862)

[ГЛАВА 4. Логика приложения. 12](#_Toc451342863)

[4.1 Добавление заметки. 12](#_Toc451342864)

[4.2 Удаление заметки. 12](#_Toc451342865)

[4.3 Редактирование заметки. 12](#_Toc451342866)

[ГЛАВА 5. Пользовательский интерфейс приложения. 12](#_Toc451342867)

[5.1 Расположение элементов. 12](#_Toc451342868)

[5.2 Фон. 12](#_Toc451342869)

[ЗАКЛЮЧЕНИЕ. 12](#_Toc451342870)

[СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ. 13](#_Toc451342871)

# ВВЕДЕНИЕ

Современный высокотехнологичный мир предлагает множество уникальных приложений на различные устройства, которые способны улучшить качество повседневной жизни. Деловые программы типа “Персональный органайзер” или “Календарь” могут стать незаменимыми помощниками в различных ситуациях, когда возникает необходимость быстро записать важную информацию на своем устройстве или предварительно распланировать дела, что позволяет упростить рабочий процесс и повысить плодотворность деятельности.

Целью данной курсовой работы являлась разработка расширения “To-Do List” для браузера Google Chrome, с удобным и понятным пользовательским интерфейсом. В настоящее время браузер - это чуть ли не самая часто используемая компьютерная программа, поэтому существует возможность быстро и легко расширять его возможности, делая пребывание в Интернете еще более комфортным.

Для достижения цели данной курсовой работы было необходимо решить следующие задачи:

* Изучить особенности разработки расширения для браузера Google Chrome и его структуру.
* Реализовать авторизацию с помощью аккаунта Google+.
* Реализовать взаимодействие данного расширения с Google Drive пользователя с помощью Google Drive API.
* Продумать логику и смоделировать интерфейс расширения.

После запуска расширения в браузере в правом верхнем углу экрана появляется окно, запускается функция авторизации с помощью Google+ аккаунта и появляется анимация ожидания. Если авторизация прошла успешно (то есть пользователь уже авторизован в данном браузере), то анимация ожидания пропадает и всплывает окно с приветствием. Если же пользователь еще не авторизован, то появляется скрытая кнопка Authorize. В случае, если пользователь авторизовался в расширении первый раз – появится запрос на разрешение доступа к личным данным профиля. После успешного завершения авторизации, расширение полностью готово к использованию.

# ГЛАВА 1. Google Chrome extension.

## 1.1 Понятие и характерные черты Google Chrome расширения.

В настоящее время большинство людей используют свои компьютеры в основном для выхода в Интернет, поэтому браузеры являются наиболее часто запускаемыми программами. Для более комфортной работы пользователей существует возможность создания расширения, поскольку создатели браузеров предоставляют всю необходимую информацию для разработки расширений.

Расширение для браузера — это компьютерная программа, работающая в связке с обозревателем и наделяющая его новыми возможностями. Терминология при этом в разных программах может отличаться. Они могут назваться дополнениями (add-on), плагинами (plug-in), расширениями (extension). Очень важно отличать эти термины, которые по сути являются синонимами, но имеют тонкие семантические различия между собой.

Плагин – независимо компилируемый програмный модуль, динамически подключаемый к основной программе, предназначенный для расширения или, в основном, использования ее возможностей. Также может переводиться как «модуль».

Аддон – добавление различного дополнительного материала (в основном используется для компьютерных игр).

Расширение – используется для повышения функциональности браузера, изменения интерфейса, добавления кнопок, панелей и других графических элементов. Расширения вносят новые возможности в браузер. На самом базовом уровне расширение Chrome – это набор HTML, CSS, JavaScript-файлов, позволяющий добавить некоторую функциональность в Chrome через JavaScript API, который предоставляет браузер. По сути, расширение – это веб-страница в Google Chrome, имеющая доступ к некоторым дополнительным API. Расширения также могут работать только на определенных страницах, выполняться в фоне или модифицировать содержание страницы. [[1]](#_СПИСОК_ИСПОЛЬЗОВАННОЙ_ЛИТЕРАТУРЫ.)

## 1.2 Создание Google Chrome extension и заполнение manifest.json.

Первое, что необходимо сделать, это создать базовую структуру проекта и папку со всеми необходимыми файлами, такими как .html, .css, .js, .ico. Типичное расширение для Chrome состоит из файла манифеста и некоторой комбинации из фоновой страницы, страниц пользовательского интерфейса и сценариев. Все начинается с файла манифеста с именем manifest.json. Этот файл содержит информацию, необходимую Chrome для того, чтобы загрузить расширение — заголовок, описание, необходимые разрешения, иконки и т.п. [[2]](#_СПИСОК_ИСПОЛЬЗОВАННОЙ_ЛИТЕРАТУРЫ.)

Для начала необходимо заполнить обязательные поля в файле манифеста:

{

"manifest\_version": 2,

"name": " To-Do List",

"description": "This extension will show To-Do List",

"version": "1.0"

}

Далее - рекомендуемые поля:

“default\_locale”: “en”; // язык по умолчанию

“icons”: {…},

“short\_name”: ”Short Name”;

Используются “разрешения” на действия:

"permissions": [

"identity",

"https://accounts.google.com/o/oauth2/token"

]

"oauth2": {

"client\_id": "121706767932-v6bt4jgv4kn3295p46kk46255g237njv.apps.googleusercontent.com",

"scopes": [

"https://www.googleapis.com/auth/drive",

"https://www.googleapis.com/auth/plus"

]

}

Content Security Policy (CSP, политика защиты контента) — это механизм обеспечения безопасности, с помощью которого можно защищаться от атак с внедрением контента. CSP описывает безопасные источники загрузки ресурсов, устанавливает правила использования встроенных стилей, скриптов. Для работы с Google API в Chrome extension необходимо внести скрипт https://apis.google.com/ в список разрешенных скриптов с ключевым словом self (соответствует текущему источнику). [[3]](#_СПИСОК_ИСПОЛЬЗОВАННОЙ_ЛИТЕРАТУРЫ.)

"content\_security\_policy": "script-src 'self' https://apis.google.com/; object-src 'self'"

Установка иконки приложения:

"browser\_action": {

"default\_icon": "icon.png"

}

Чтобы протестировать разработанное расширение необходимо подключить его в браузере. Для этого надо перейти в **Меню Chrome -> Инструменты -> Расширения** и загрузить распакованное расширение. Если файл манифеста будет содержать ошибку – то расширение будет отключено или просто не загрузится. Установленный значок в виде иконки должен появиться на правой панели. После этого можно начинать работу с расширением.

Также существует возможность обратно упаковать расширение для дальнейшего его распространения. Chrome запросит каталог с распакованным расширением и файл секретного ключа. Упаковав расширение, его можно распространять любым способом — по электронной почте, через свой Web-сайт, посредством программы установки и т.п.

# ГЛАВА 2. Авторизация.

## 2.1 Открытый OAuth протокол.

Не так уж давно, каждый веб-сервис требовал отдельной регистрации, но все большая часть из них дает возможность пользователю использовать для авторизации свой аккаунт в социальных сетях. Реализуется эта возможность благодаря протоколу OAuth. Это открытый протокол для авторизации, который позволяет предоставить третьей стороне ограниченный доступ к защищённым ресурсам пользователя без необходимости передавать ей (третьей стороне) логин и пароль.

При использовании OAuth-протокола пользователь не передает свой логин и пароль к защищенным ресурсам напрямую в приложение. Поэтому у пользователя больше оснований доверять приложению, поскольку пользователь может быть уверен, что несанкционированный доступ к его личным данным невозможен. Не владея логином и паролем пользователя, приложение сможет выполнять только те действия с данными, которые разрешил пользователь, и никакие другие. Так же при разработке приложения не нужно заботиться об обеспечении конфиденциальности логина и пароля пользователя, например, шифруя его. [[4]](#_СПИСОК_ИСПОЛЬЗОВАННОЙ_ЛИТЕРАТУРЫ.)

Перед инициализацией протокола, разработчик должен зарегистрироваться на сервере авторизации, предоставляя информацию о типе клиента, URL-адресе перенаправления (куда он должен быть перенаправлен для авторизации, после того, как сервер ресурса выдаст ему разрешение или же откажет в доступе) и любую другую информацию, необходимую серверу.

Взамен он получает уникальный идентификатор клиента (**client\_id**) и секретный код клиента (**client\_secret**). Этот процесс называется регистрацией клиента. После регистрации клиент сможет взаимодействовать с сервером.

## 2.2 Регистрация приложения в Google API Console.

Для разработки приложения с поддержкой авторизации с использованием открытого протокола OAuth, необходимо зарегистрировать приложение в Google API Console. Это делается через регистрационную форму в Google Console, где необходимо предоставить информацию о своём приложении (название приложения, сайт приложения (или id Google Chrome расширения), URL-адрес перенаправления). [[5]](#_СПИСОК_ИСПОЛЬЗОВАННОЙ_ЛИТЕРАТУРЫ.)

Как только приложение будет зарегистрировано, сервер будет выдавать учетные данные клиента в виде идентификатора клиента и секретного кода клиента. Идентификатор клиента - это публичная строка, которая используется API сервиса для идентификации приложения, а также используется для построения авторизационных URL’ов, которые представляются пользователям. Секретный код клиента используется для установки подлинности приложения для API сервиса, когда приложение запрашивает доступ к аккаунту пользователя и должен быть сохранен в секрете между приложением и API.

На основе этих данных, будет сформирована ссылка кода авторизации, которая выглядит следующим образом:

<https://account.google.com/o/oauth/auth?response_type=code&client_id=CLIENT_ID&redirect_uri=CALLBACK_URL&scope=https://www.googleapis.com/auth/userinfo.email>

* https://account.google.com/o/oauth/auth — конечная точка авторизации
* response\_type = code — предусматривает что приложение запрашивает предоставление кода авторизации
* client\_id = CLIENT\_ID — идентификатор клиента приложения (значение, по которому API определяет (идентифицирует) приложение).
* redirect\_uri = CALLBACK\_URL — место, куда сервис перенаправляет пользователя после того, как предоставлен код авторизации.
* Scope = read — определяет уровень доступа.

## 2.3 Авторизация через Google.

Процесс авторизации состоит из трех взаимосвязанных частей:

* function handleClientLoad() { window.setTimeout(checkAuth, 1); } – вызывается, когда приложение полностью загружено и готово начать процесс авторизации.
* function checkAuth() {

gapi.auth.authorize({'client\_id': CLIENT\_ID, 'scope': https://www.googleapis.com/auth/drive, 'immediate': true}, handleAuthResult);

} – проверка, авторизован ли текущий пользователь в приложении.

* function handleAuthResult(authResult) {

if (authResult && !authResult.error) {

gapi.client.load('drive', 'v2', function () {

getFile(); - функция для получения файла с google drive.

});

} else {

authButton.onclick = function () {

gapi.auth.authorize(

{ 'client\_id': CLIENT\_ID, 'scope': SCOPES[0], 'immediate': false}, handleAuthResult);

};

} – функция вызывается, если сервер авторизации ответил на запрос.

# ГЛАВА 3. Работа с Google Drive API.

## 3.1 Описание Google Drive API.

Google предлагает широкий спектр интерфейсов прикладного программирования, чаще всего базирующихся вокруг актуальных продуктов Google. Интерфейс прикладного программирования – это интерфейс создания приложений, основанный на использовании готовых решений (например, функций).

Google API – это набор инструментов (клиентская библиотека), позволяющий создавать приложения, работающие с данными сервисов Google, в случае Google Drive API, с данными Google Drive пользователя.

При разработке приложения, разработчик может использовать данный API для реализации взаимодействия программы и Google Drive пользователя, обмена данными между ними. Таким образом, при работе с программой, конечный пользователь сможет либо вручную сохранять данные на Диске, которые после этого будут доступны на любых других устройствах, либо может быть реализовано автоматическое сохранение (например, через определенный промежуток времени). [[6]](#_СПИСОК_ИСПОЛЬЗОВАННОЙ_ЛИТЕРАТУРЫ.)

## 3.2 Работа с Google Drive.

В данной программе реализовано несколько функций для работы c Google Drive:

* function createFile()
* function updateFile(fileId)
* function findFile(fname)
* function downloadFile(file)
* function checkFileExists(fname, callback)

# ГЛАВА 4. Логика приложения.

## 4.1 Добавление заметки.

## 4.2 Удаление заметки.

## 4.3 Редактирование заметки.

# ГЛАВА 5. Пользовательский интерфейс приложения.

## 5.1 Расположение элементов.

## 5.2 Фон.

# ЗАКЛЮЧЕНИЕ.

# СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ.

1. Создание расширения для Google Chrome - <http://programmict.ru/delaem-rasshirenie-dlya-google-crome/>
2. Manifest File Format - <https://developer.chrome.com/extensions/manifest>
3. Content Security Policy - <https://habrahabr.ru/company/nixsolutions/blog/271575/>
4. OAuth tutorial - <https://developer.chrome.com/extensions/tut_oauth>
5. Creating a Google Developers Console project and client ID - <https://developers.google.com/identity/sign-in/web/devconsole-project>
6. JavaScript QuickStart Drive API - <https://developers.google.com/drive/v3/web/quickstart/js>