МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ  
Федеральное государственное бюджетное   
образовательное учреждение высшего образования   
ВЯТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Факультет автоматики и вычислительной техники  
Кафедра систем автоматизации управления

Методические указания   
по выполнению практического задания 1  
по дисциплине  
Web-технологии   
  
для студентов по направлениям подготовки   
09.03.02 Информационные системы и технологии

Киров 2023

# Цели и задачи

Цель работы**:** исследование технологий создания Progressive Web Application PWA приложения на стеке технологий NodeJS.

# Порядок выполнения работы

## Теоретические основы построения PWA приложений

Объяснение преподавателя о технологии PWA.

## Организация рабочего окружения.

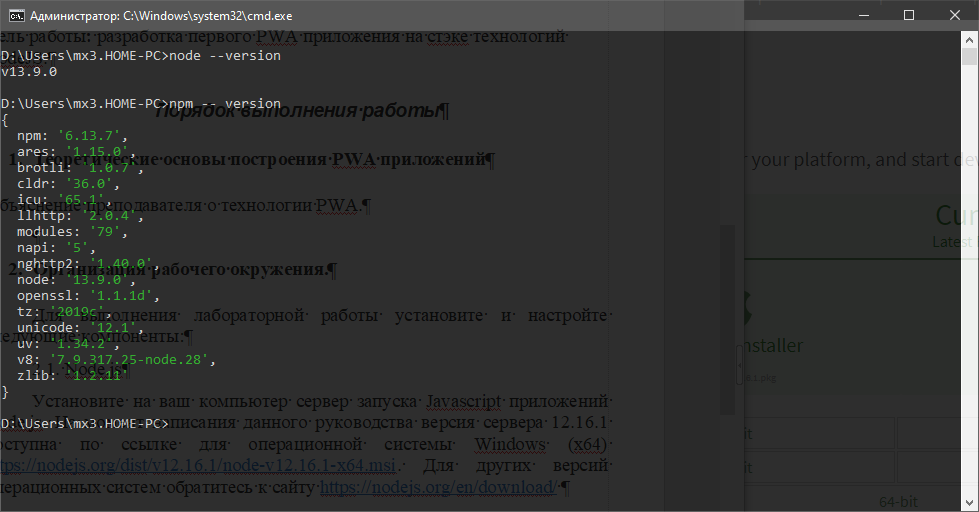
Для выполнения лабораторной работы установите и настройте следующие компоненты:

2.1. Node.js

Установите на ваш компьютер сервер запуска Javascript приложений Node.js. На момент написания данного руководства версия сервера 12.16.1 доступна по ссылке для операционной системы Windows (x64) <https://nodejs.org/en/download/>. Для удобства работы можно использовать nvm (https://github.com/nvm-sh/nvm ).

Одновременно вместе с установкой сервера Node.js будет установлен менеджер пакетов npm.

Протестируйте работу сервера и менеджера пакетов узнав их версии.



Для дальнейшей работы изучите простое введение в команды работы с менеджером npm <http://prgssr.ru/development/vvedenie-v-paketnyj-menedzher-npm-dlya-nachinayushih.html>

Настройке npm для работы с глобальными настройками в директории, позволяющей запись модулей. Проверьте работу npm установив глобально модуль colors

npm install -g colors

2.2. Google Chrome

Установите браузер google chrome для дальнейшей работы с PWA.

Lighthouse - это автоматизированный инструмент с открытым исходным кодом для повышения его качества. Lighthouse встроен в Google DevTools. Изучите работу инструмента Lighthouse <https://developers.google.com/web/tools/lighthouse/>.

2.3. Vue

Современные веб-приложения построены на модульных JavaScript библиотеках. Одной из таких библиотек является Vue.js.

Для работы с Vue существует модуль Vue CLI, который можно установить командой



Внимательно изучите методы работы с утилитой vue. Используйте официальную документацию для изучения методов работы с утилитой.

Какие инструменты есть в утилите?

Какие проекты позволяет создавать утилита?

2.4. Workbox

Workbox - это набор библиотек и модулей Node.js, которые упрощают кеширование ресурсов и в полной мере используют возможности, используемые для создания PWA. Чтобы узнать больше о Workbox, перейдите на страницу <https://developers.google.com/web/tools/wokbox/>. Установите глобально его в вашей системе:



2.5. Firebase

Firebase - это облачный сервис, который помогает автоматизировать задачи разработки бэкенда. Firebase - это место, где вы можете сохранять данные и наборы данных, проводить авторизацию пользователей без понимания серверной разработки. Необходимо установить Firebase CLI :



В дальнейшем потребуются настройки на стороне сервиса Firebase

[https://console.firebase.google.com](https://console.firebase.google.com/)

2.6. git

Проверьте наличие git и аккаунта для github. Все работы сдаются только в виде ссылки на репозиторий github.com. Создайте репозиторий

ISN\_2020\_{#вариант от 1 до 33}.

Создайте файл описания Readme.md в ветке master. Ветка master будет хранить все отчеты по практическим заданиям. Ссылки на отчеты должны содержаться в Readme .md.

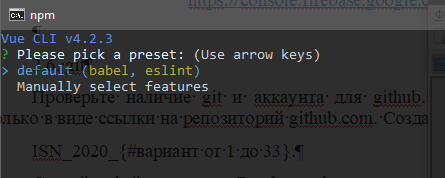
Сами задания будем сохранять в соответствующие ветки Work#N, где N- номер выполняемой работы.

## Разработка PWA приложения

1. Создайте директорию с проектами с правами rwrwrw и откройте в ней консоль терминала. Создание приложения Vue начинается с команды create. Создайте проект приложения isn-app:

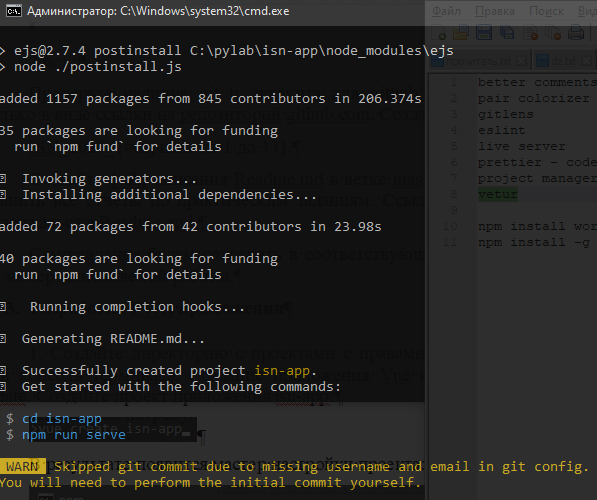
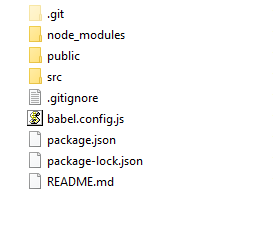


В результате появится мастер настройки проекта:



Выбираем все настройки default.

В результате выполения команды будет сформирован готовый минимальный проект рабочего приложения vue

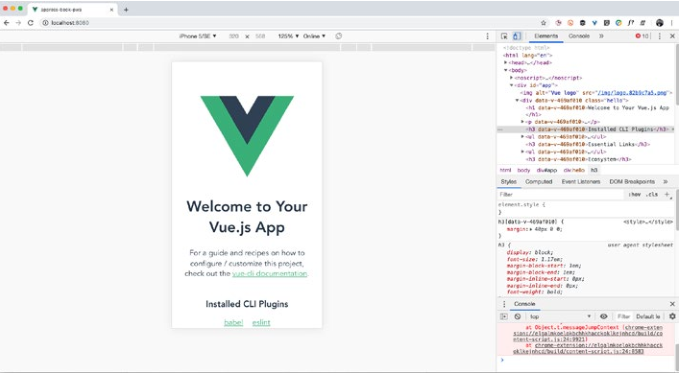
 

2. Протестируйте работу базового приложения выполнив последовательно команды

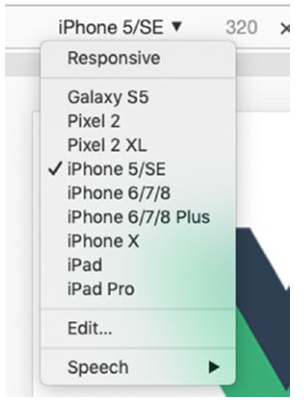


3. Изучите работу клиентской части данного приложения. Проведите аудит. Какие ошибки показывает аудит?

Изучите работу приложения в режиме отображения на мобильном устройстве.



Выберите в качестве целевого устройства iPhone 5/SE



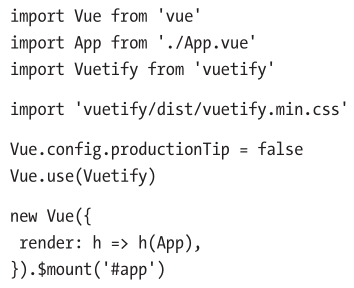
Какие проблемы вы обнаружили при таком отображении приложения?

Для более аккуратной поддержки устройств с разными разрешениями установим поддержку декоратора Material Design (тема Material Degign используется Google в мобильных приложениях, поэтому такой веб-сайт будет стилизован под мобильные приложения).

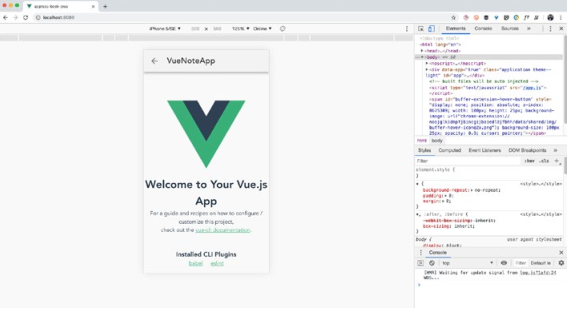
4. Подключим тему Material Design путем добавления компонента во vue:



Подключим модуль в файле приложения main.js:



Проверим работу полученного приложения в браузере Google Chrome при мобильном отображении контента.

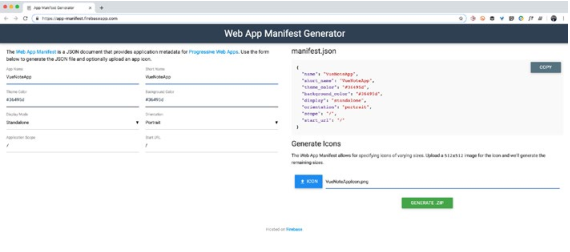


5. Сгенерируйте манифест приложения, используя сервис

<https://app-manifest.firebaseapp.com/>

При генерации манифеста используйте любую иконку размером 512х512 px.

Пример работы генератора представлен ниже:



В результате должен быть получен следующий конфиг

{

"name": "ISNApp",

"short\_name": "ISNApp",

"theme\_color": "#36495d",

"background\_color": "#36495d",

"display": "standalone",

"orientation": "portrait",

"Scope": "/",

"start\_url": "/",

"icons": [

{

"src": "./icons/icon-72x72.png",

"sizes": "72x72",

"type": "image/png"

},

{

"src": "./icons/icon-96x96.png",

"sizes": "96x96",

"type": "image/png"

},

{

"src": "./icons/icon-128x128.png",

"sizes": "128x128",

"type": "image/png"

},

{

"src": "./icons/icon-144x144.png",

"sizes": "144x144",

"type": "image/png"

},

{

"src": "./icons/icon-152x152.png",

"sizes": "152x152",

"type": "image/png"

},

{

"src": "./icons/icon-192x192.png",

"sizes": "192x192",

"type": "image/png"

},

{

"src": "./icons/icon-384x384.png",

"sizes": "384x384",

"type": "image/png"

},

{

"src": "./icons/icon-512x512.png",

"sizes": "512x512",

"type": "image/png"

}

],

"splash\_pages": null

}

При нажатии на кнопку «Generate ZIP» вы скачаете все ресурсы в в сжатом файле. Распакуйте и вставте в папку проекта. Далее нам нужно сообщить веб-приложению, что существует файл манифеста. Добавьте следующую строку в index.html:

<link rel="manifest" href="/manifest.json">

5. Проверьте получившийся результат. Файл манифеста должен быть скачен при на локальный комптютер.

Изучите отображение файла манифеста в Chrome DevTools

6. Service Worker

Service Worker- это сценарий (файл JS), который выполняется в фоновом режиме, независимо от того, открыта ли веб-страница. Он добаляет такие функции, как автономная поддержка и push-уведомления.

Для создания нашего сервисного работника мы используем библиотеку под названием Workbox, который является преемником sw-toolbox и sw-precache.

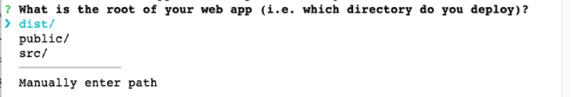
Проверьте работу Workbox на вашем компьютере, выполнив команду



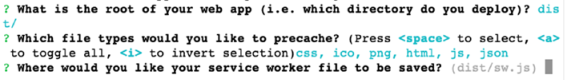
Если Workbox работает корректно запустите мастер генерации Service Worker в корневой директории проекта

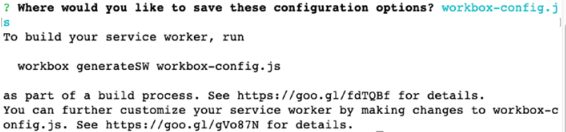


Сгенерируйте файл sw.js в директории public/

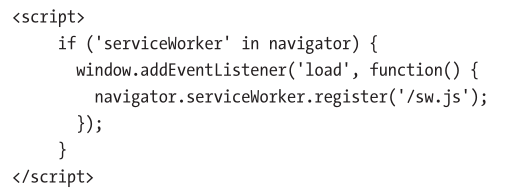


В мастере выбираем следующие ответы:





Для подключения к приложению добавьте код в файл public/index.html



Для корректной работы проекта необходимо выполнить



И затем



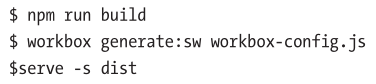
Для работы в режиме встроенного сервера разработки установите пакет serve



Запустите сервер разработки из корневой директории проекта

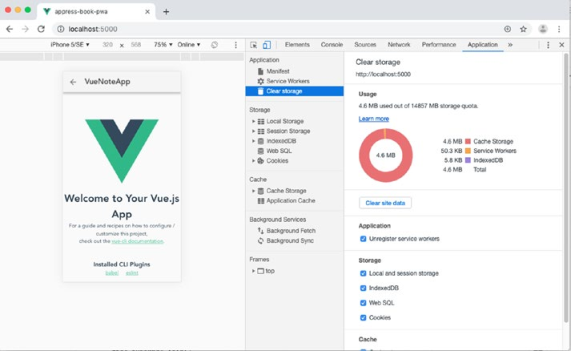


Таким образом каждый раз внося изменения в проект мы должны перекомпилировать ServiceWorker, поэтому команды запуска выглядят так:



7. Проверьте работу полученного приложения

В результате работы проверьте работу ServiceWorker



Изучите файл sw.js

8. Практические результаты

1. Репозиторий github.com с простейшим приложением Vue и отчетом по его созданию.

2. Демонстрация работы преподавателю простейшего проекта