

Практика развертывания web-приложения с помощью Docker

1. Подготовка к созданию приложения и установка компонент

Наше приложение будет состоять из 2 частей **frontend** и **backend**, которые будут обмениваться между собой с помощью http-запросов. Мы будем создавать приложения с использованием js-фреймворка **vuejs** 3 версии и с использованием фреймворка **Django**. Для создания веб-приложения с использованием этих технологий нам необходимы следующие инструменты, перед непосредственной разработкой установим их.

1.1 Установка **python**

На **python** в нашем приложении будет реализован бэкенд.

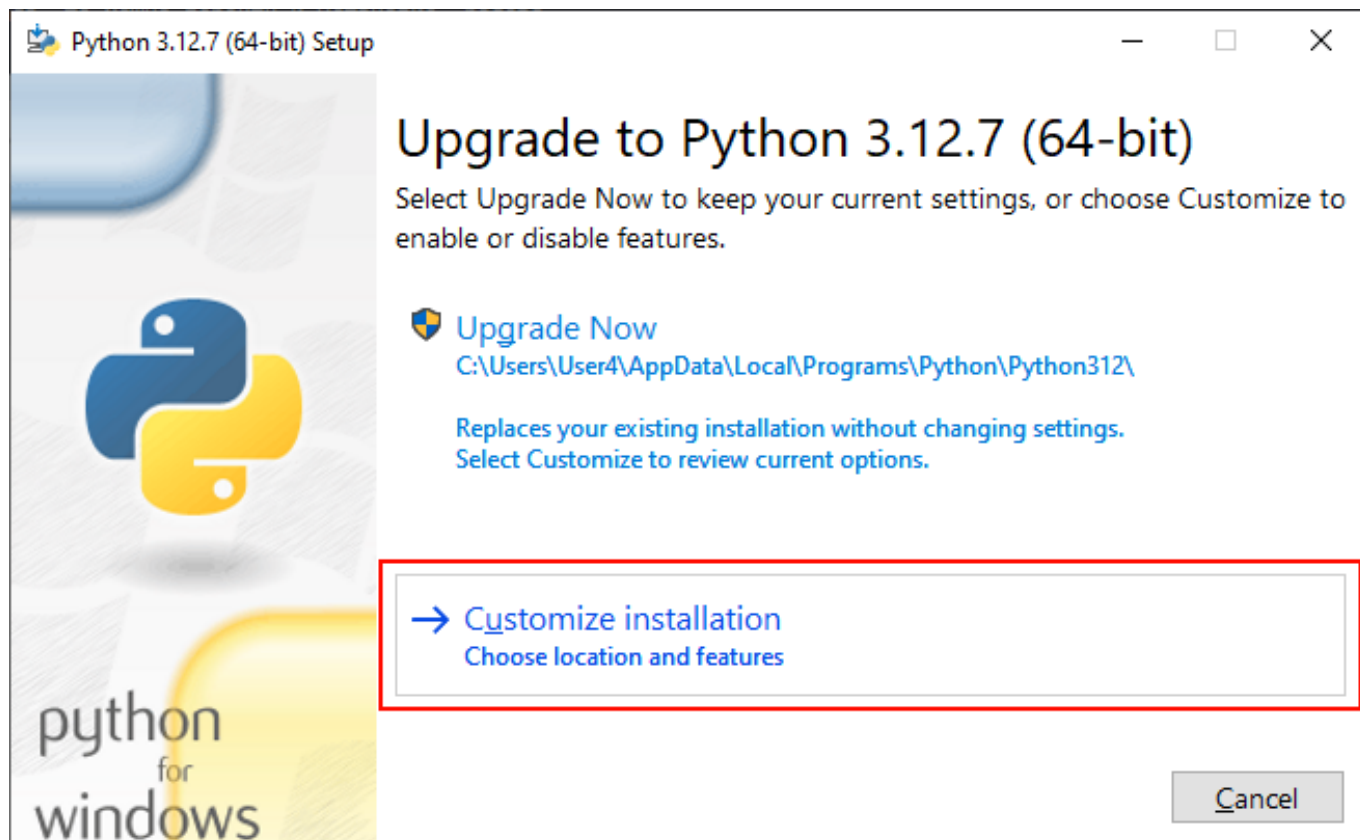
Сначала проверим установлен ли **python** или нет. Для этого в командной строке введем следующую команду

```
python -V
```

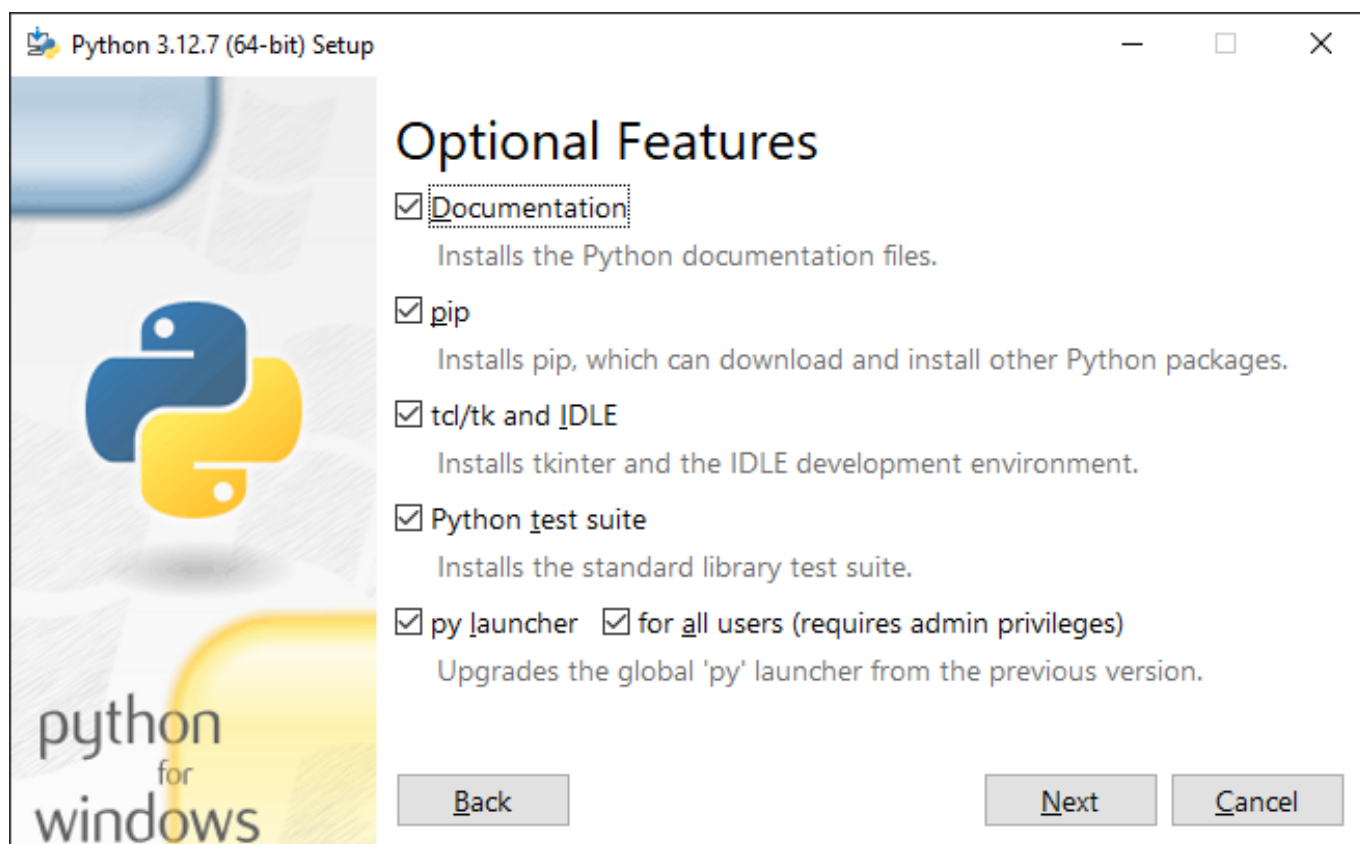
Если команда успешно отработала, то можно перейти к установке **nodejs**.

Если **python** не установлен, то идем на сайт и качаем python, либо сразу по [ссылке](#) внизу выбираем нужную версию os и скачиваем.

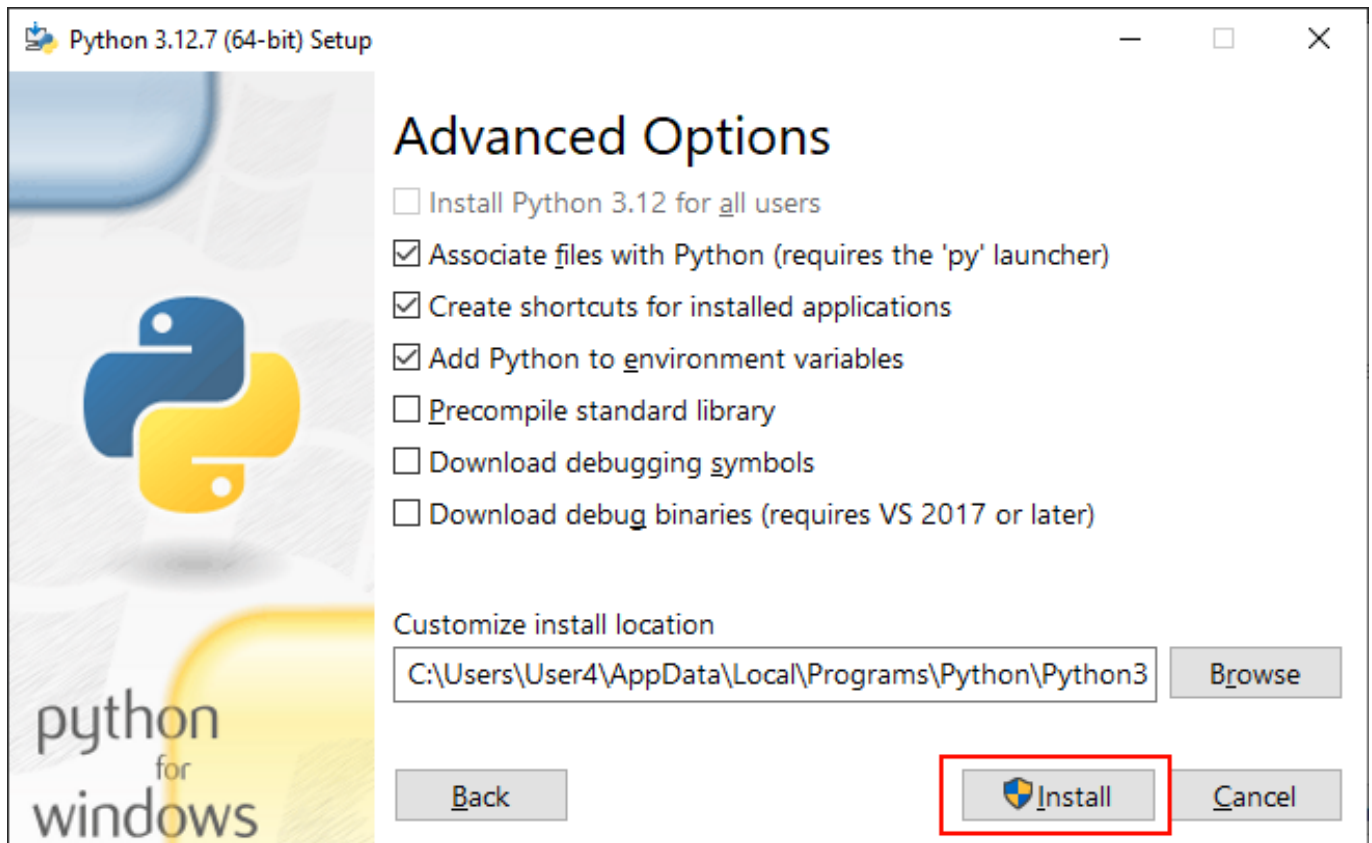
После скачивания открываем установщик и выбираем кustomную установку



Затем оставляем все по умолчанию



Идем дальше и устанавливаем



После чего, проверяем в командной строке что все успешно установилось

```
python -V
```

Вывелась версия, значит все ок и идем дальше!

1.2 Установка Node JS

Реализовывать фронтенд часть нашего приложения мы будем с помощью фреймворка **VueJS** для него необходимо посадить "ноду" (она же **nodejs**). В кратче это платформа, с помощью которой можно запустить приложение написанное на javascript.

Javascript - это тоже язык программирования. Отличный материал по введению что это можно открыть [здесь](#).

Проверим, установлена ли "нода" выполним команду

```
npm -v
```

Данной командой мы проверяем установлен ли пакетный менеджер **nodejs**

Если выводится версия, то все хорошо, скачивать и устанавливать его не нужно. Если нет то идем ставить.

На официальном [сайте](#) выбираем скачать, скорее всего версия 22.11. После чего запускаем установщик, все оставляем по умолчанию.

После установки проверяем что все установилось также командой

```
npm -v
```

Установим пакет для создания **vuejs** приложений следующей командой

```
npm install -g @vue/cli
```

2. Организация приложения

Для структуризации будет создавать приложение в папке **todo-app**

Важно! Необходимо чтобы полный путь до директории **todo-app** не содержал в себе символов кириллицы (т.е. например "C:\work\todo-app" - ОК, а "C:\пользователь232\todo-app" нет)

3. Создание фронтенд части приложения

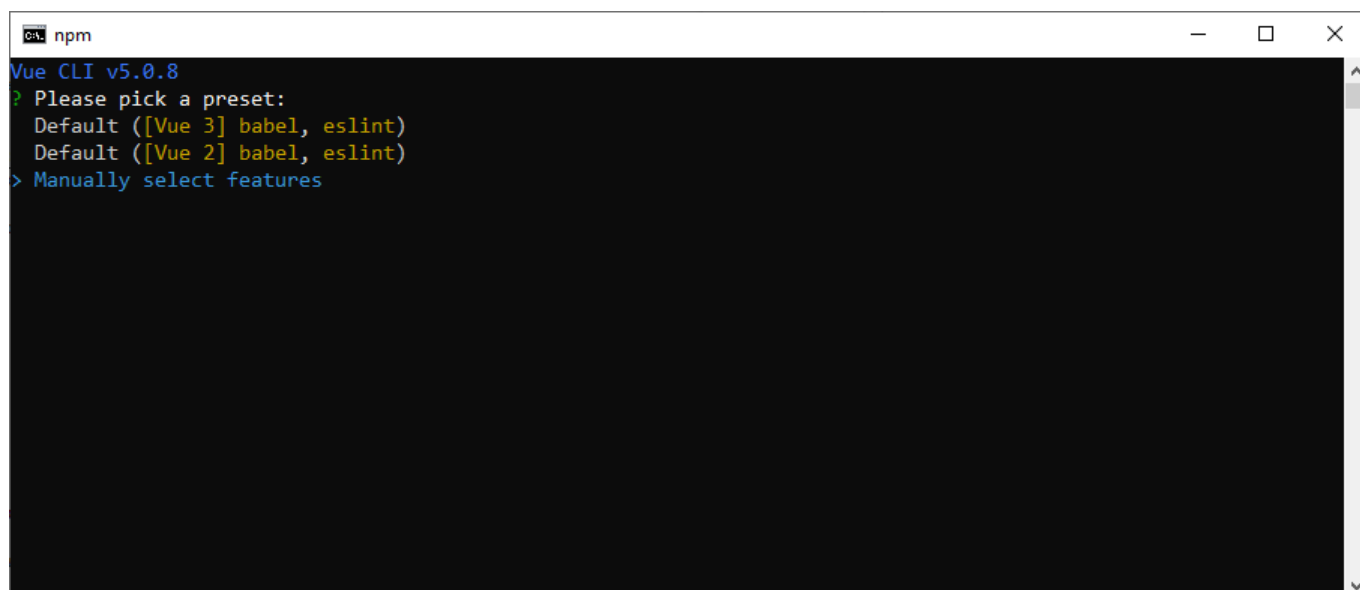
3.1 Создание vue проекта

Для создания **vue** приложения **перейдем в папку todo-app** и откроем командную строку ней.

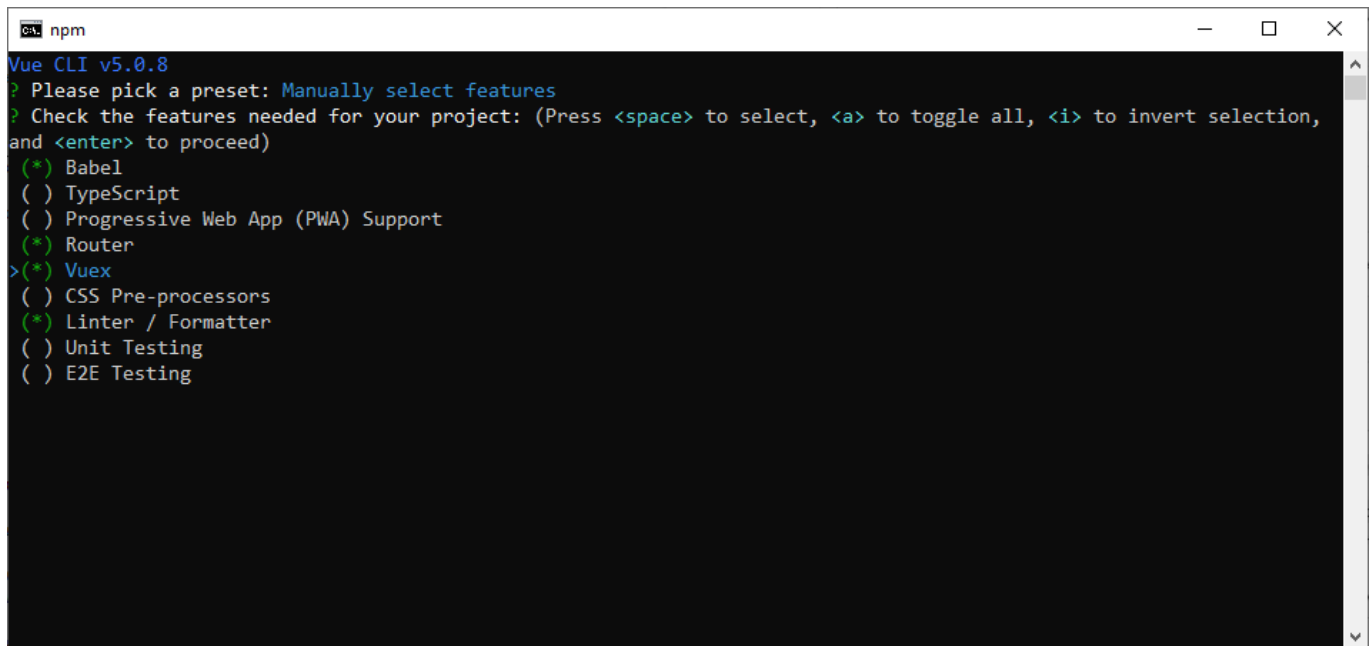
В командной строке создаем vue приложение командой

```
vue create frontend
```

В консоли откроется менеджер создания приложения выберем ручную установку (3 пункт выбираем и жмем **Enter**)

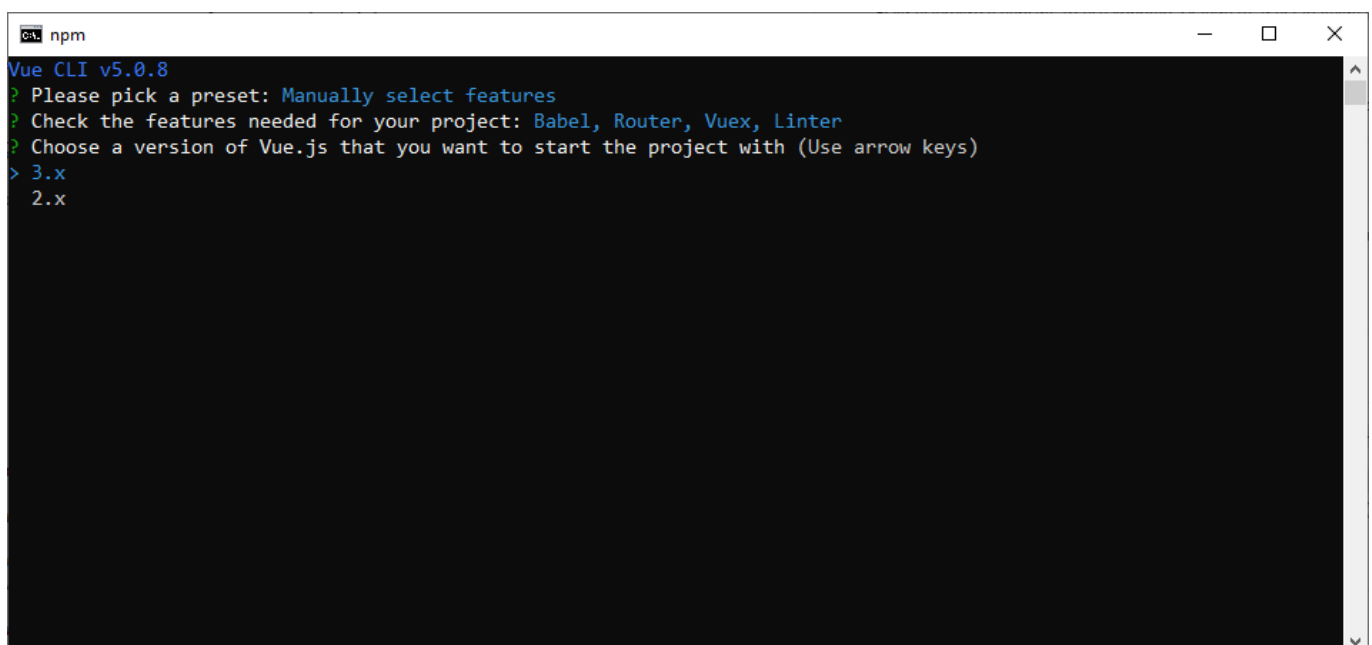


На следующей шаге выберем настройки нашего проекта нажимая "ПРОБЕЛ"



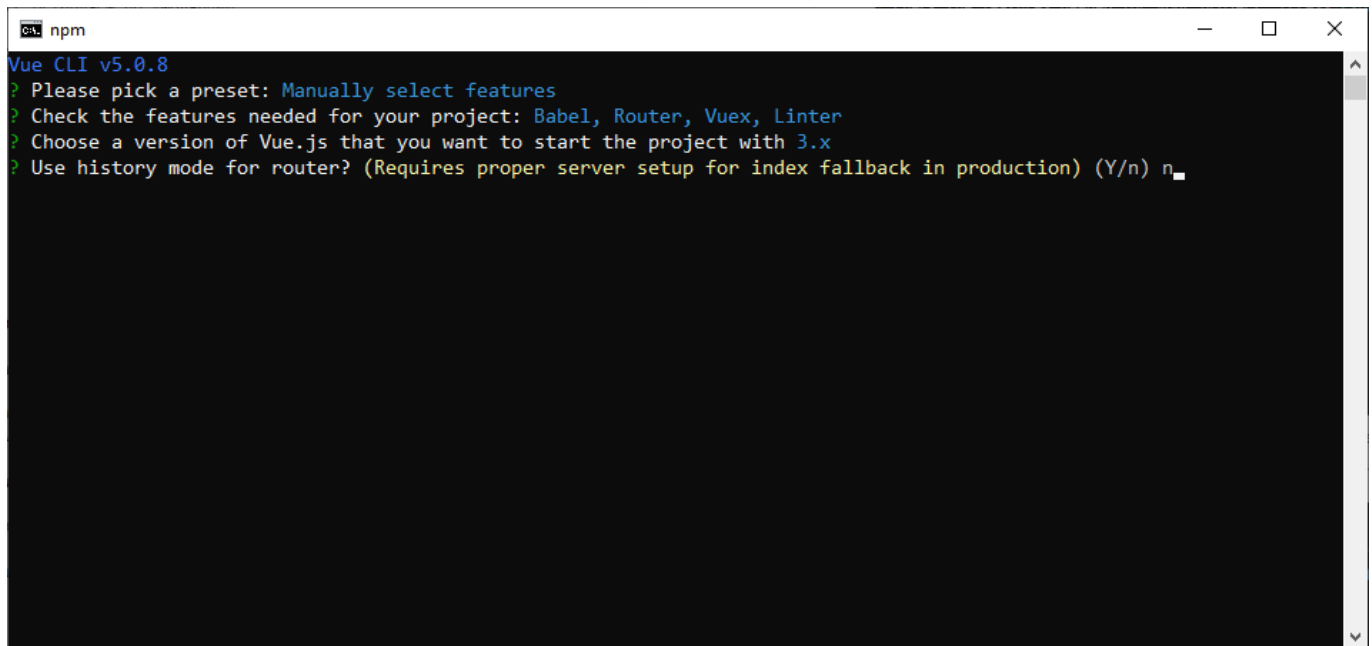
```
npm
Vue CLI v5.0.8
? Please pick a preset: Manually select features
? Check the features needed for your project: (Press <space> to select, <a> to toggle all, <i> to invert selection,
and <enter> to proceed)
(*) Babel
( ) TypeScript
( ) Progressive Web App (PWA) Support
(*) Router
> (*) Vuex
( ) CSS Pre-processors
(*) Linter / Formatter
( ) Unit Testing
( ) E2E Testing
```

После выбора, нажимаем **Enter** - переходим к следующему шагу с выбором версии и выбираем 3 версию



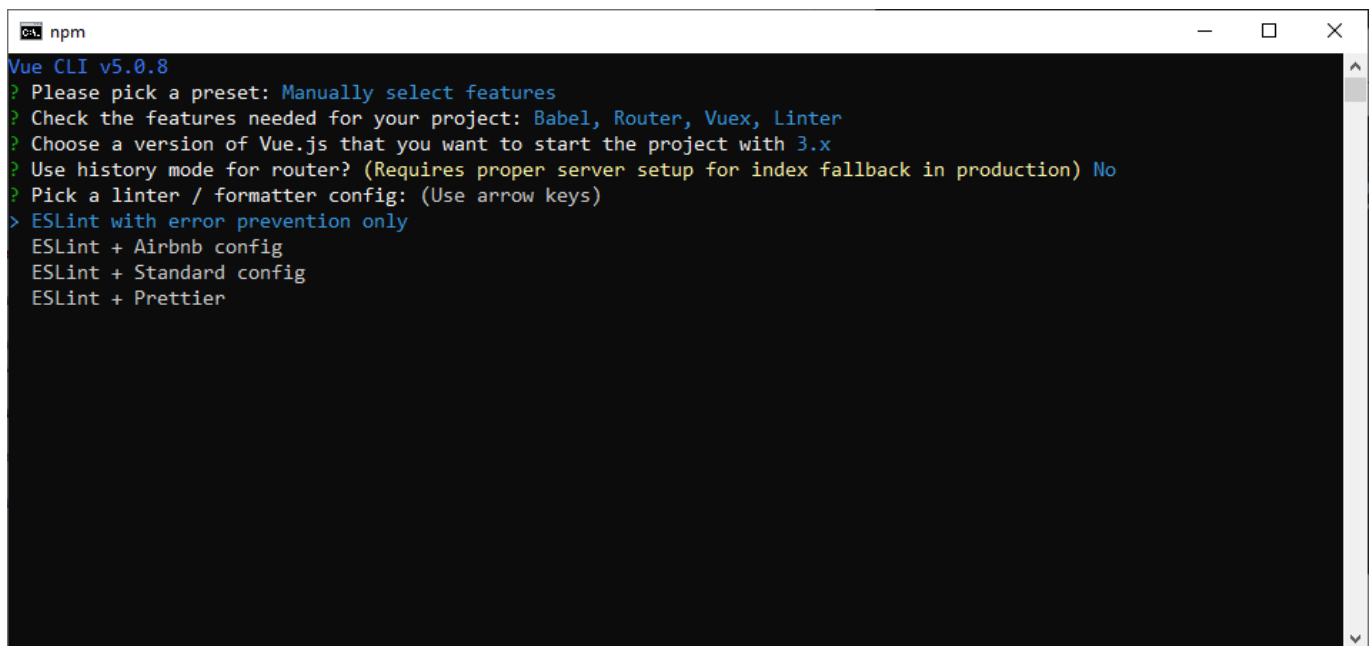
```
npm
Vue CLI v5.0.8
? Please pick a preset: Manually select features
? Check the features needed for your project: Babel, Router, Vuex, Linter
? Choose a version of Vue.js that you want to start the project with (Use arrow keys)
> 3.x
  2.x
```

На следующем шаге выбираем **n**



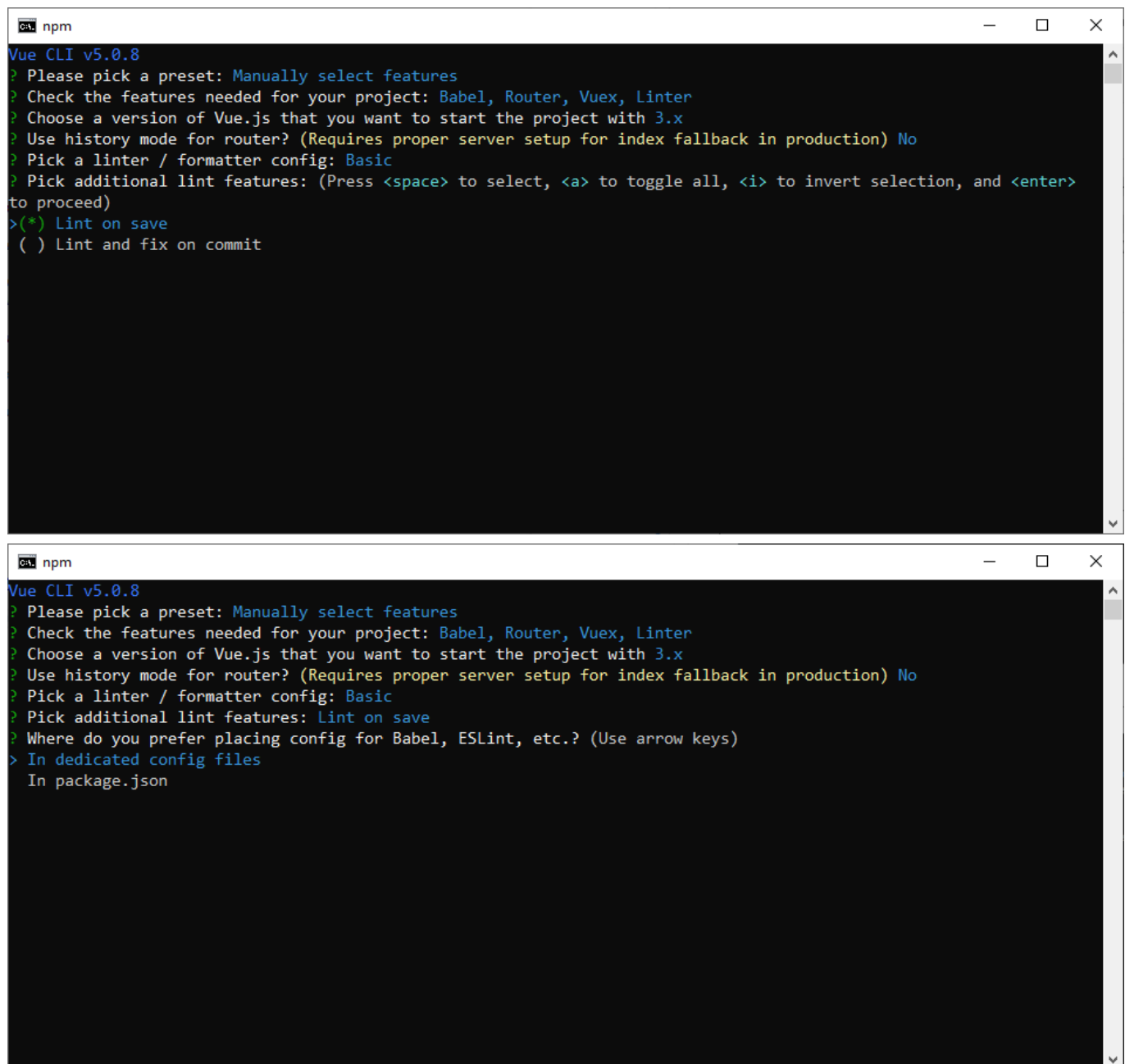
```
npm
Vue CLI v5.0.8
? Please pick a preset: Manually select features
? Check the features needed for your project: Babel, Router, Vuex, Linter
? Choose a version of Vue.js that you want to start the project with 3.x
? Use history mode for router? (Requires proper server setup for index fallback in production) (Y/n) n
```

Дальше по умолчанию



```
npm
Vue CLI v5.0.8
? Please pick a preset: Manually select features
? Check the features needed for your project: Babel, Router, Vuex, Linter
? Choose a version of Vue.js that you want to start the project with 3.x
? Use history mode for router? (Requires proper server setup for index fallback in production) No
? Pick a linter / formatter config: (Use arrow keys)
> ESLint with error prevention only
  ESLint + Airbnb config
  ESLint + Standard config
  ESLint + Prettier
```

Снова по умолчанию



The image shows two sequential screenshots of a terminal window running the npm CLI v5.0.8. The window title is 'npm'. The first screenshot shows the initial prompts: 'Please pick a preset: Manually select features', 'Check the features needed for your project: Babel, Router, Vuex, Linter', 'Choose a version of Vue.js that you want to start the project with 3.x', 'Use history mode for router? (Requires proper server setup for index fallback in production) No', 'Pick a linter / formatter config: Basic', and 'Pick additional lint features: (Press <space> to select, <a> to toggle all, <i> to invert selection, and <enter> to proceed)'. The second screenshot shows the next prompts: '>(*) Lint on save', '() Lint and fix on commit', and 'Where do you prefer placing config for Babel, ESLint, etc.? (Use arrow keys)'. The user has selected 'In dedicated config files'.

```
npm
Vue CLI v5.0.8
? Please pick a preset: Manually select features
? Check the features needed for your project: Babel, Router, Vuex, Linter
? Choose a version of Vue.js that you want to start the project with 3.x
? Use history mode for router? (Requires proper server setup for index fallback in production) No
? Pick a linter / formatter config: Basic
? Pick additional lint features: (Press <space> to select, <a> to toggle all, <i> to invert selection, and <enter> to proceed)
>(*) Lint on save
( ) Lint and fix on commit

npm
Vue CLI v5.0.8
? Please pick a preset: Manually select features
? Check the features needed for your project: Babel, Router, Vuex, Linter
? Choose a version of Vue.js that you want to start the project with 3.x
? Use history mode for router? (Requires proper server setup for index fallback in production) No
? Pick a linter / formatter config: Basic
? Pick additional lint features: Lint on save
? Where do you prefer placing config for Babel, ESLint, etc.? (Use arrow keys)
> In dedicated config files
  In package.json
```

На последнем шаге выбираем **n** - не сохраняем пресет создания проекта

```
npm
Vue CLI v5.0.8
? Please pick a preset: Manually select features
? Check the features needed for your project: Babel, Router, Vuex, Linter
? Choose a version of Vue.js that you want to start the project with 3.x
? Use history mode for router? (Requires proper server setup for index fallback in production) No
? Pick a linter / formatter config: Basic
? Pick additional lint features: Lint on save
? Where do you prefer placing config for Babel, ESLint, etc.? In dedicated config files
? Save this as a preset for future projects? (y/N) n
```

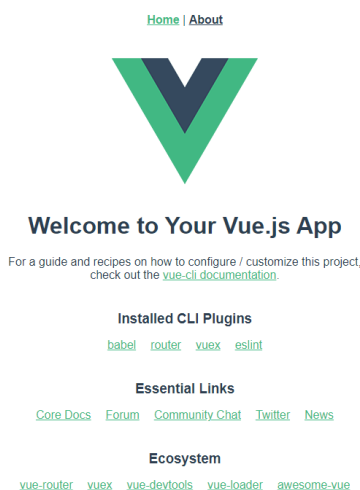
Нажимаем **Enter** и ждем пока создается проект



Когда создание завершится - в папке **todo-app** создаться каталог **frontend** с нашим фронтенж приложением, мы можем его запустить выполнив следующие команды:


```
cd frontend
npm run serve
```

Последней командой мы как раз запускаем наше приложение, которое можно посмотреть перейдя по адресу <http://localhost:8080/>. Должно получиться следующее



Если все ок, то можно вернуться в терминал и остановить сервер командой **Ctrl+C**

3.2 Добавление библиотек и фреймворков (tailwindcss, axios)

3.3 Добавление шаблонов для страниц

4. Создание бэкенда

Для бэкенда мы будем использовать **Django** фреймворк. Но сперва создадим в директории **todo-app** каталог **backend**, перейдем в него и откроем из этой папки командную строку.

4.1 Создание виртуального окружения

Виртуальное окружение нужно для того, чтобы библиотеки, используемые в нашем текущем проекте не конфликтовали с другими будущими проектами. Некоторые версии библиотек python не дружат с другими версиями, создание виртуального окружения под свой проект является хорошей практикой.

Существует несколько способов создания виртуального окружения. Самый простой с помощью библиотеки **python**.

Установим библиотеку для создания виртуальных окружений с помощью пакетного менеджера **pip**.

```
pip install virtualenv
```

С помощью это команды мы устанавливаем библиотеку в глобальное хранилище библиотек **python**.

Затем в директории **backend** нашего приложения **todo-app** создаем виртуальное окружение командой.

```
python -m virtualenv .venv
```

После чего в каталоге **backend** должна появиться папка **.venv** в которой будет содержаться интерпретатор и набор библиотек (модулей).

4.2 Создание проекта Django

Создадим наше приложение **Django**. Для это сперва **активируем наше виртуальное окружение** которое мы создали под наш проект.

Выполним следующую команду (Для не Windows команды может отличаться см. активировать venv + ВАША_ОС)

```
.venv\Scripts\activate
```

Слева в терминале должно появиться (**.venv**) - что говорит о том, что мы успешно активировали виртуальное окружение. Теперь мы можем устанавливать различные библиотеки в него, которые не будут мешать и конфликтовать с модулями из глобального хранилища библиотек.

Установим **Django** фреймворк в наше виртуальное окружение, воспользуемся все тем же пакетным менеджером **pip**, только теперь модули будут ставится не глобально а наше окружение **.venv**, т.к. оно активировано у нас сейчас.

```
pip install django
```

Теперь мы можем создать django приложение, для этого выполним следующую команду (находясь в папке **backend**)

```
django-admin startproject backend .
```

После чего должны появиться каталог **backend** (его называют пакет конфигурации) и файл **manage.py** (это модуль - обертка над командами джанго)

Мы создали стартовое приложение **django**, можем попробовать его запустить, для это как раз вызовем модуль **manage.py** и передадим ему параметр **runserver**

```
manage.py runserver
```

Теперь перейдем по адресу <http://127.0.0.1:8000/>, где будет запущена дефолтная страница фреймворка django



The install worked successfully! Congratulations!

View [release notes](#) for Django 5.1

You are seeing this page because `DEBUG=True` is in your settings file and you have not configured any URLs.

django



Django Documentation
Topics, references, & how-to's



Tutorial: A Polling App
Get started with Django



Django Community
Connect, get help, or contribute

Если все ок, то можем вернуться в консоль и остановить сервер командой **Ctrl+C**

4.3 Добавление логики приложения

5 Фронтенд + бэкенд + postgres

6 Создание Docker-образов

6.1 Docker образ для фронтенда

6.2 Docker образ для бэкенда

7 Оркестрация контейнеров с использованием docker compose