

# ***1. Хостинг веб-приложения на сервисе Heroku***

Целью данной лабораторной работы является включение имеющейся клиентской части (лабораторных работ прошлого семестра) внутрь нового приложения, которое будет развернуто в сервисе облачного хостинга Heroku.

Контрольный срок сдачи работы: 4-ая неделя обучения.

Для выполнения данной лабораторной работы вам необходимо:

- 1) [Установить NodeJS](#) (LTS-версия);
- 2) [Установить пакет NestJS CLI](#) глобально через пакетный менеджер;
- 3) Зарегистрироваться в хостинг-сервисе [Heroku](#);
- 4) Установить [Heroku CLI](#).

Чтобы создать приложение NestJS с помощью Nest CLI, необходимо выполнить команду `nest new` в терминале вашей операционной системы. Укажите имя проекта и выберите пакетный менеджер, которым вы пользуетесь (по умолчанию npm). В результате будет сгенерирован проект в новой директории с именем вашего проекта.

Откройте проект в IDE и ознакомьтесь с файлом `package.json`. Отредактируйте поле `author` согласно [схеме](#), укажите ваше авторство.

Проверьте, что ваше приложение работает, для этого выполните скрипт запуска, описанный в `package.json` (`npm run start` в случае, если вы пользуетесь пакетным менеджером по умолчанию).

Обратите внимание, что запущенное приложение запускается на 3000-ом порту: <http://localhost:3000>. Для хостинга Heroku такое поведение не подходит, т.к. Heroku сообщает вам через переменную окружения с именем `PORT` номер порта, на котором необходимо принимать соединения. Биндинг вашего приложения на 3000-ый порт осуществляется в файле `src/main.ts`. Отредактируйте файл таким образом, чтобы приложение начинало слушать подключения на порту, указанному в переменной окружения, а в случаях, когда порт не указан, брать порт по умолчанию (на ваш выбор).

Переменные окружения хоста могут быть прочитаны двумя способами:

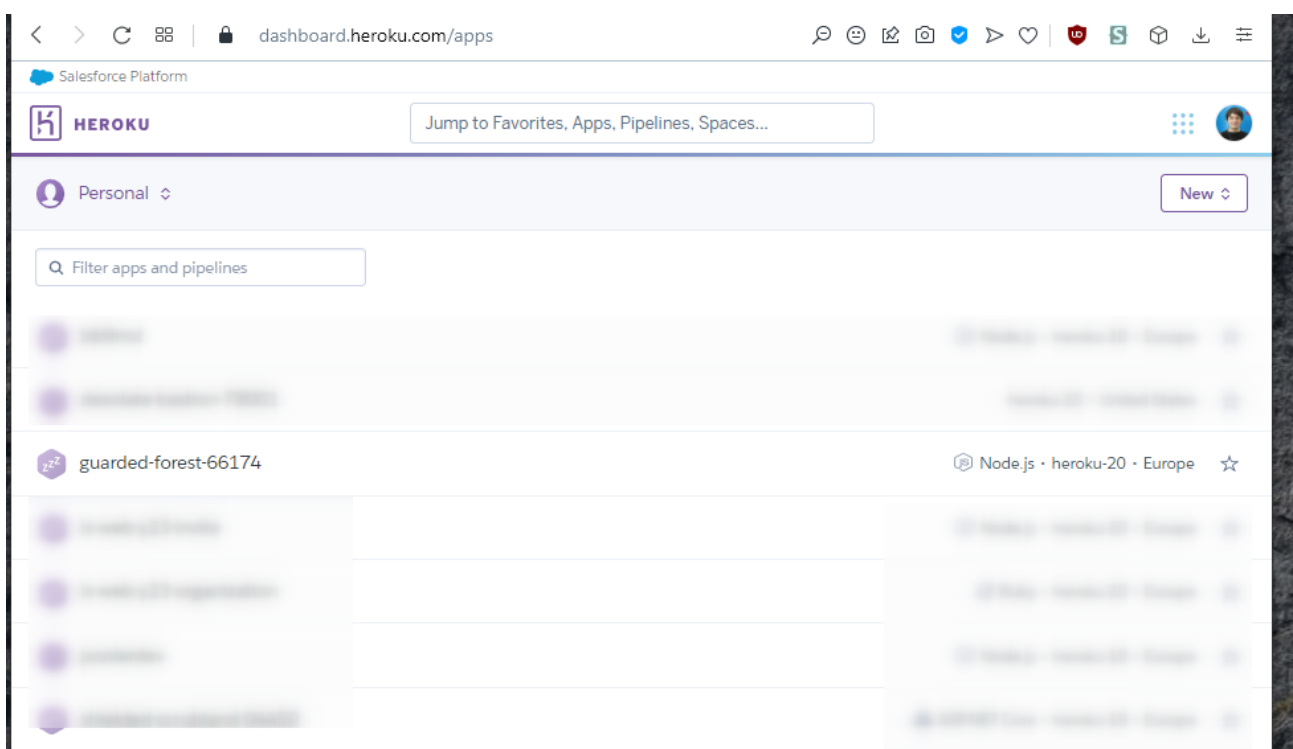
- 1) Напрямую средствами NodeJS через [process.env](#);
- 2) С помощью [модуля конфигурации Nest](#).

После добавления необходимой обработки проверьте, что приложение стартует на указанном в переменной окружения порту, для этого необходимо настроить профиль запуска ([пример для WebStorm](#)).

Если всё получилось, то можно переходить к следующему шагу — созданию самого инстанса приложения на хостинге. Выполните команду `heroku apps:create --help` в терминале вашей ОС и ознакомьтесь с параметрами. Укажите название приложения и выберите регион Europe (eu).

Остальные параметры можно оставить по умолчанию. Внимание: Название приложения будет использовано в качестве поддомена.

Убедится, что приложение создано, можно во вкладке Personal Dashboard на облачном хостинге Heroku.



Следующим шагом необходимо указать входную точку (предоставить команду, с помощью которой должно запускаться ваше приложение) [в файле Procfile](#).

Сервис Heroku будет создавать по одному исполняющему процессу на каждую строчку в этом файле. Среди всех типов исполняющих процессов вам необходимо указать тип `web` для запуска непосредственно веб-приложений и предоставить команду для запуска.

В качестве входной точки рекомендуется использовать production-версию вашего приложения, т.е. команду `npm run start:prod`.

После создания **Procfile** необходимо загрузить ваш код на сам хостинг.

Heroku умеет осуществлять автоматическую сборку и запуск приложений через средства Git. Для этого необходимо инициализировать репозиторий и связать его с Heroku, для этого можно воспользоваться Heroku CLI, выполнив команду `heroku git:remote`.

После того, как вы добавите все файлы вашего приложения и запустите код на хостинг, должен [начаться процесс сборки](#) с отображением прогресса непосредственно в консоли Git.

Если все прошло без ошибок, то ваше приложение должно быть доступно в Интернете, выполните команду `heroku apps:open`, чтобы получить перманентную ссылку, и откройте её в браузере.

Отлично, осталось только разместить статические ресурсы (на данный момент статикой будет являться весь фронтенд, который был разработан в рамках лабораторных работ прошлого семестра) и научиться отдавать их с сервера. Для этого необходимо воспользоваться [следующим рецептом](#).

Укажем в момент запуска нашего Nest-приложения, что мы хотим воспользоваться шаблоном MVC, основанном на фреймворке [Express](#). Укажем, что хотим иметь возможность отдавать статические ресурсы, расположенные в конкретной папке (по умолчанию `public`). Затем необходимо будет перенести все ваши файлы с frontend'ом в указанную папку и проверить, что статика отдаётся, т.е. сделать запрос на <http://localhost:port/index.html>.

Делаем коммит и пушим результат.

После проделанных шагов можно считать лабораторную работу выполненной, остается только предоставить ваши результаты.

Как будет построен дальнейшей процесс сдачи:

- В вашем репозитории в GitHub-организации (об организации см. ниже) создайте ветку с номером лабораторной работы (вида `lab-1`) и загрузите туда актуальный код (тот же `master`, который сейчас находится на Heroku);
- Создайте pull request вашей ветки в мастер, подойдите в порядке живой очереди **очно**, предоставьте практику ссылку на pull request и устно защитите ваше решение.

Как создать репозиторий в организации и для чего это вообще нужно?

Так как Heroku, которым вы пользуетесь, персональный, а для доступа в ваш репозиторий непосредственно на хостинге необходима корпоративная учетная запись, мы будем хранить ваш код в двух репозиториях, на GitHub и на Heroku.

На момент сдачи допускается, что код может отличаться, но

подразумевается, что по просьбе преподавателя практики у вас будет возможность загрузить на Heroku код из ветки на GitHub.

Как попасть в организацию?

Для этого необходимо заполнить форму (<https://is-web-y23-invite.herokuapp.com>) и предоставить ссылку на ваше приложение, это нужно, чтобы преподаватели могли отслеживать ваш прогресс в течение семестра. После авторизации вы сразу же получите приглашение в организацию.

Следующие лабораторные работы ищите в публичном репозитории внутри организации :)