Курс NextJS. Эскиз контейнеризации и развертывания

Д. Макаренков, к.т.н.

https://dmpsy.club

NextJS >= ReactJS + NodeJS

Целевая аудитория

Энтузиасты программирования на NextJS, желающие следовать современным принципам разработки full-stack Web-приложений

Магическая формула:

NextJS > = ReactJS (front-end) + NodeJS (back-end)

План работы / Agenda

1	Containerization & deployment of the NextJS app. Statement of the problem	Контейнеризация и развертывание приложения NextJS. Постановка задачи
2	Creation and testing of the standalone NextJS app image	Создание и тестирование образа standalone-приложения NextJS
3	Two-container NextJS (Web app. + DB schema) app architecture	Целевая 2-контейнерная (Web -прил. + схема БД) архитектура приложения NextJS
4	Configuring and testing (NextJS App + PostgreSQL + pgAdmin) environment with docker compose	Настройка и тестирование стенда: NextJS App + postgreSQL + pgAdmin под управлением docker compose
5	Context (C1), container (C2), component (C3) and environment deployment diagrams	Диаграммы контекста (C1), контейнеров (C2), компонента (C3) и развертывания среды
6	Improving the current environment implementation	Развитие текущей реализации стенда
7	Summary	Подведем итоги

Контейнеризация и развертывание приложения NextJS. Постановка задачи

В прошлой лекции мы обсудили основные свойства статической отрисовки (static rendering). Сегодня посмотрим, как это проявляется при контейнеризации приложения в Docker

За основу возьмем “официальный” Dockerfile из примеров NextJS и применим его для создания образа нашего приложения nexxt-app: сначала без подключения к БД, а затем с коннектом к PostgreSQL, запущенной в отдельном контейнере, исходя из требований микросервисной архитектуры и модели C4

Для запуска как отдельных контейнеров, так и всей среды создадим docker-compose.yml - файлы, что упростит последующее развертывание приложения в Kubernetes, docker swarm и т.д.

В качестве standalone-приложения берем вариант из лекции №4, а с соединением с PostgreSQL - вариант со статическим рендерингом из прошлой лекции

Подготовительные шаги

Prepare both nodejs and pnpm	**Подготовьте nodejs и pnpm, установите fnm:** **curl -o- https://fnm.vercel.app/install | bash** **fnm install 22** **fnm list** *** v22.14.0 default** *** system** **node -v** **npm -v # Should print "10.9.2"** **npm install pnpm -g** **pnpm -v # Should print "10.8.0"**
Verify that docker installed	**Убедитесь, что docker установлен и свеж:** **docker -v** **Docker version 28.1.1, build 4eba377** **Обновите или установите, при необходимости**

Создаем образ приложения (без коннекта к БД)

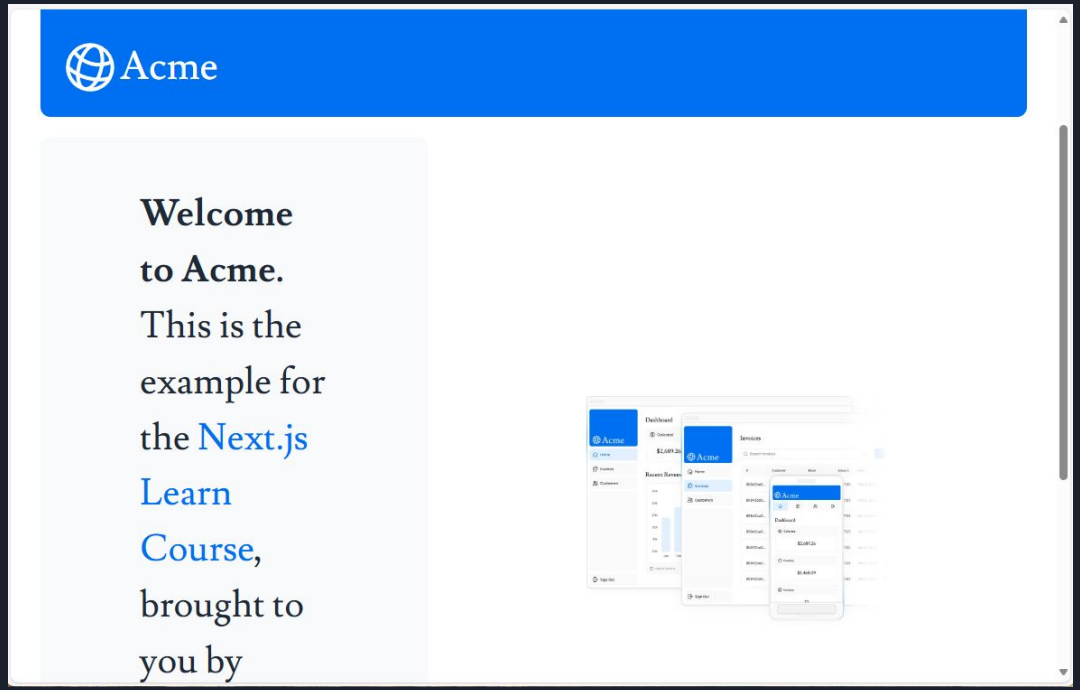
1	Clone the application	Клонируйте проект и перейдите в его папку: git clone https://github.com/DmitryMakar/nextjs-lesson-004.git cd nextjs-lesson-004
2	Install node modules	Установите необходимые пакеты (node modules): pnpm install Если потребуется, подтвердите установку доп. пакетов: pnpm approve-builds
3	Build the image	Раскомментируйте output: “standalone” в next.config.mjs Соберите образ nexxtapp:latest docker build -t nexxtapp .
4	Verify image creation	Убедитесь, что образ nexxtapp:latest создан: docker images REPOSITORY TAG IMAGE ID CREATED SIZE nexxtapp latest 8d1970ccd9fd About a minute ago 181MB

Запускаем контейнер с приложением nexxtapp

5	**Launch the nexxtapp container**	**Запустите контейнер в режиме detached на порту 3000** **docker run -p 3000:3000 -d nexxtapp**
6	**Check that the container has started**	**Убедитесь, что контейнер стартовал:** **docker container ls -a** **CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES** **7257f6e86e69 nexxtapp "docker-entrypoint.s…" 25 seconds ago Up 22 seconds 0.0.0.0:3000->3000/tcp, [::]:3000->3000/tcp modest_mestorf**

Запускаем контейнер с приложением nexxtapp (2)

7	**Open the website**	**Проверьте, что приложение открывается:** **http://localhost:3000/** **http://<virtual_machine_IP>:3000/ #если запустились на VM**



Завершаем работу с приложением nexxtapp

8	Stop the nexxtapp container	Остановите контейнер по его ID или имени: docker container stop 7257f6e86e69 docker container stop modest_mestorf
9	Remove the container	Удалите контейнер: docker container rm modest_mestorf
10	Remove the image, when necessary	При необходимости, удалите образ nexxtapp: docker image rm nexxtapp:latest Untagged: nexxtapp:latest Deleted: sha256:d996454c93ddc345df42ff91cb80df4081e8ddbef3b82b8c655a13915f016d30

Результаты контейнеризации standalone приложения nexxtapp

1. Предлагаемый универсальный Dockerfile не вполне оптимизирован в плане размера генерируемого образа и скорости его сборки
2. Есть несколько предупреждений об устаревшем синтаксисе команд
3. В целом, Dockerfile обеспечивает сборку NextJS-приложений
4. Для работы с образами удобно использовать docker-compose - файлы

```
dmitry@kubernetes:~/NextJSTraining/lesson-008d$ cat docker-compose.single-app.yml
services:
  nexxt_app:
    container_name: nexxt_container
    #image: diamondondockerhub/nextjsapp:8.0
    image: nexxtapp:latest
    ports:
      - 3000:3000
    healthcheck:
      test: ["CMD-SHELL","ls -l /tmp >> /tmp/list.txt"]
      interval: 60s
      timeout:  5s
      retries:  1
```

docker compose up # запустить среду

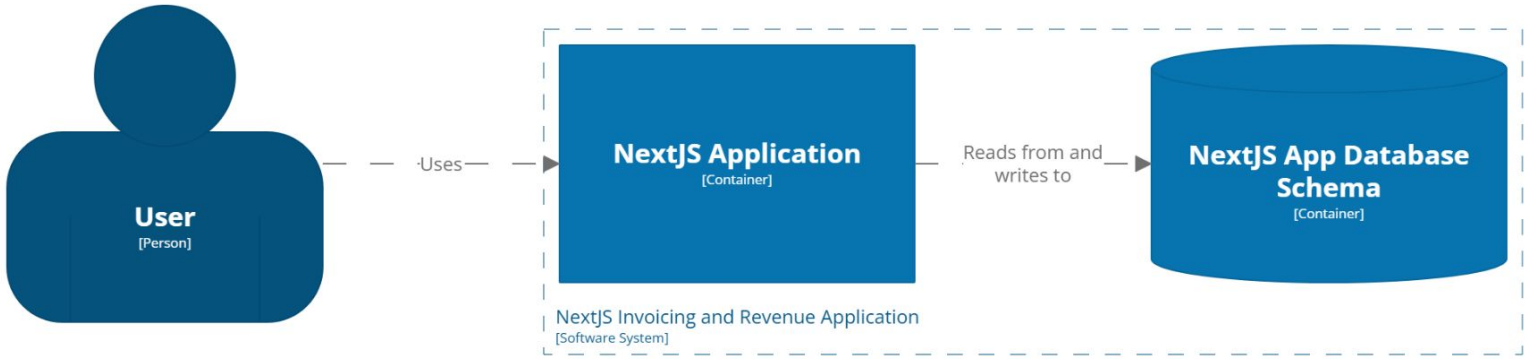
docker compose down # остановить среду и удалить ее конфигурацию

docker compose --help # вывести все доступные команды

docker compose -f docker-compose.single-app.yml up -d # запустить из файла в режиме detached

Целевая архитектура: отдельные контейнеры для приложения и БД

Исходя из принципов микросервисной архитектуры и модели C4, приложение и его схема БД должны быть разнесены по отдельным контейнерам:



[Container] NextJS Invoicing and Revenue Application

среда, 23 апреля 2025 г. в 10:46 Москва, стандартное время

Стенд для приложения NextJS с коннектом к БД PostgreSQL и pgadmin

Минимальная среда (см. Диаграмму развертывания на след. слайде) будет включать следующие компоненты:

1. Docker контейнер для размещения образа Web-приложения NextJS
2. Docker контейнер БД PostgreSQL
3. Docker контейнер утилиты pgAdmin
4. Файл инициализационных скриптов seed.mjs

Конфигурация среды задается в файле docker-compose.yaml

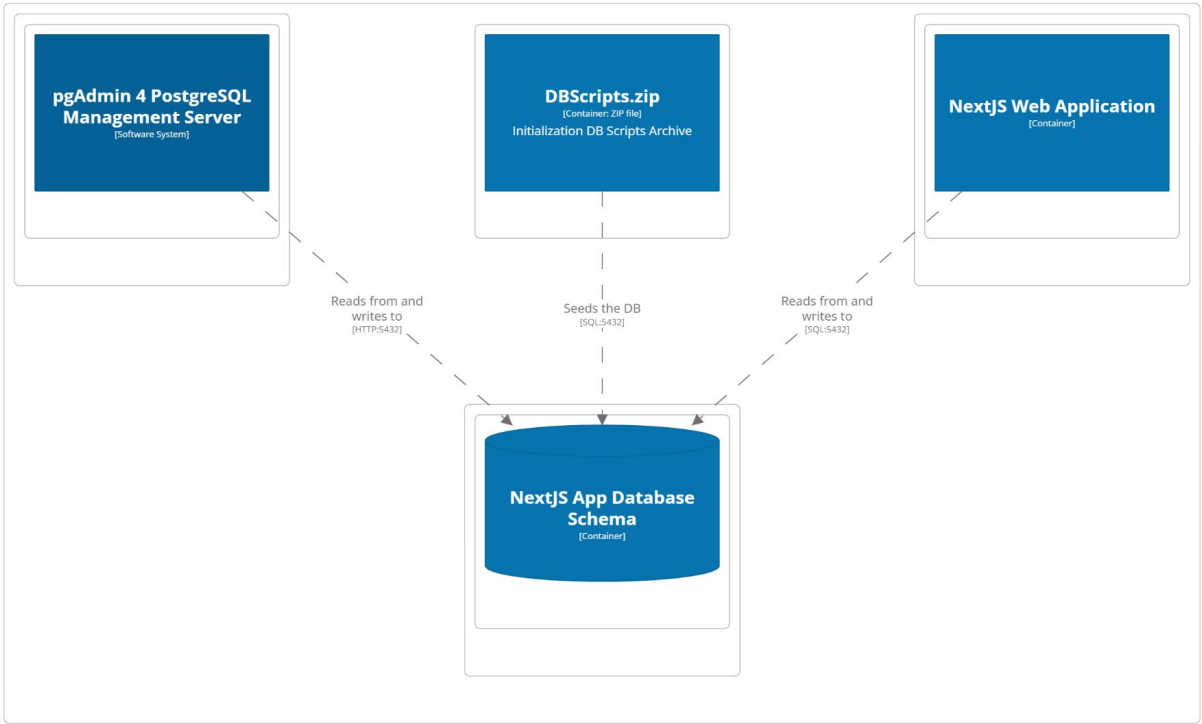
Конфигурация pgAdmin + postgreSQL адаптирована из примера:

https://github.com/LessonDump/DockerPostgresPgAdmin

Special thanks to Aleksey Voko, the author!

NB Первоначальные данные из БД уже “прошиты” в образ 8.1 при его сборке, но, в общем случае динамической отрисовки (dynamic rendering) потребуется обмен данными с БД в реальном времени, поэтому контейнер приложения NextJS стартует только после запуска контейнера БД PostgreSQL и инициализации соответствующей схемы БД (см. конфигурацию docker- compose.yaml)

Диаграмма развертывания стенда (deployment diagram)



[Deployment] NextJS Invoicing and Revenue Application - Development Environment

An example deployment environment for the NextJS application development

пятница, 25 апреля 2025 г. в 17:57 Москва, стандартное время

Разворачиваем стенд для приложения NextJS

1	Download and uncompress lesson-008d archive	Скачайте и разархивируйте lesson-008d.tar curl https://dmpsy.club/references/NextJS/lesson_008d.tar --output lesson_008d.tar mkdir lesson_008d && tar -xvf lesson_008d.tar --directory=lesson_008d
2	Launch the environment	Запустите тестовый стенд в режим “detached” cd lesson_008d mkdir pgadmin && sudo chown -R 5050:80 pgadmin # подготовьте папку docker compose up -d # запустите среду docker compose logs # см. логи запуска контейнеров

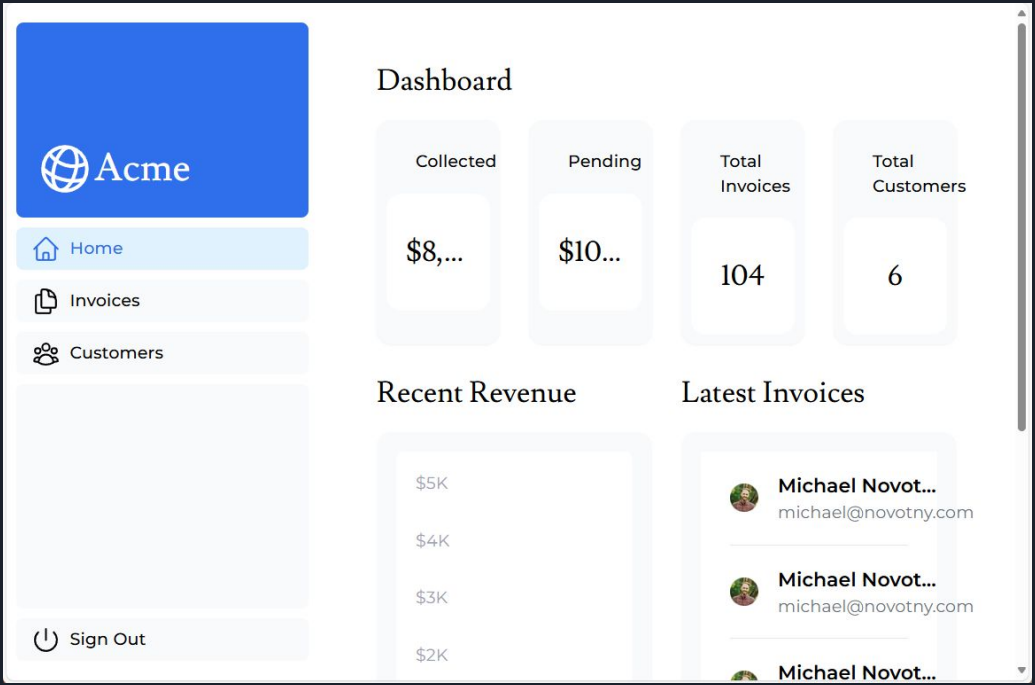
3	Seed the DB	Заполните БД исходными данными:
		cd lesson_008d & nano .env # проверьте настройки соединения
		POSTGRES_HOST=localhost # или IP-адрес хоста БД PostreSQL
		POSTGRES_USER=postgres
		POSTGRES_PASSWORD=changeme
		POSTGRES_DATABASE=postgres
		pnpm i && pnpm seed # выполните установочные скрипты

Работаем с контейнерами: NextJS (nexxt_container)

1	Check health of all containers	Удостоверьтесь, что контейнеры “healthy”, сети работают docker container ls -a docker network ls # 2 dedicated networks shown: #lesson_008d_backend #lesson_008d_postgres
2	Connect to nexxt_container and perform some validation	Зайдите в режиме командной строки: docker container exec -it nexxt_container sh Проанализируйте результаты выполнения следующих команд: ls ps -ef env netstat -al ifconfig cat /tmp/list.txt # результаты периодических healthchecks ping -c 5 postgres_container #ok ping -c 5 pgadmin_container #нет доступа Выйдите из контейнера: exit

Работаем с контейнерами: NextJS (nexxt_container) (2)

3	**Open the website**	**Проверьте, что приложение открывается:** **http://localhost:3000/dashboard** **http://192.168.17.131:3000/dashboard** **# если Host IP=192.168.17.131**



Работаем с контейнерами: PostgreSQL (postgres_container)

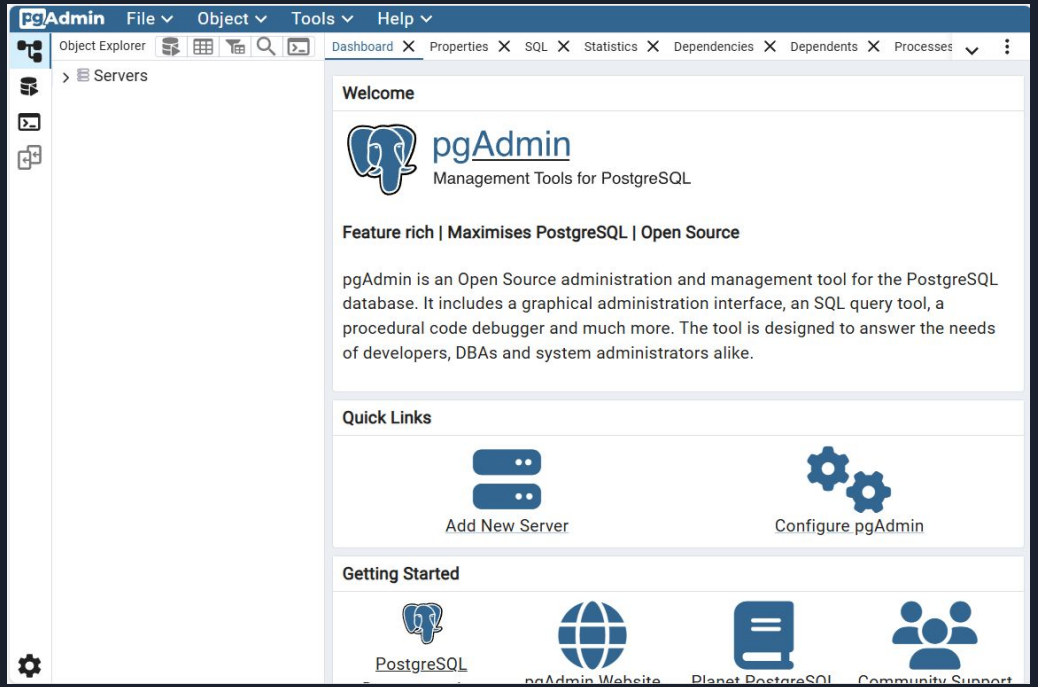
4	Connect to postgres_container and perform some validation	**Зайдите в контейнер в режиме командной строки:**
		`docker container exec -it postgres_container sh`
		Проанализируйте результаты выполнения следующих команд:
		`ls`
		`ps -ef`
		`env`
		`netstat -al`
		`lfconfig` # eth0 and eth1, 2 adapters towards 2 networks
		`ping -c 5 nexxt_container` #ok
		`ping -c 5 pgadmin_container` #ok
		`ls -l /docker-entrypoint-initdb.d` #link to the initdb folder on the host
		Выйдите из контейнера:
		`exit`

Работаем с контейнерами: pgAdmin (pgadmin_container)

5	Connect to pgadmin_container and perform some validation	**Зайдите в контейнер в режиме командной строки:**
		`docker container exec -it pgadmin_container sh`
		Проанализируйте результаты выполнения следующих команд:
		`ls`
		`ps -ef`
		`env`
		`ifconfig` # eth0, one adapter towards 1 network
		`ping -c 5 postgres_container` #ok
		`ping -c 5 next_container` #no access
		Выйдите из контейнера:
		`exit`

Работаем с контейнерами: pgAdmin (pgadmin_container) (2)

6	**Open the website**	**Проверьте, что приложение открывается:** **http://localhost:5050/browser/** **http://192.168.17.131:5050/browser/** **# если Host IP=192.168.17.131**



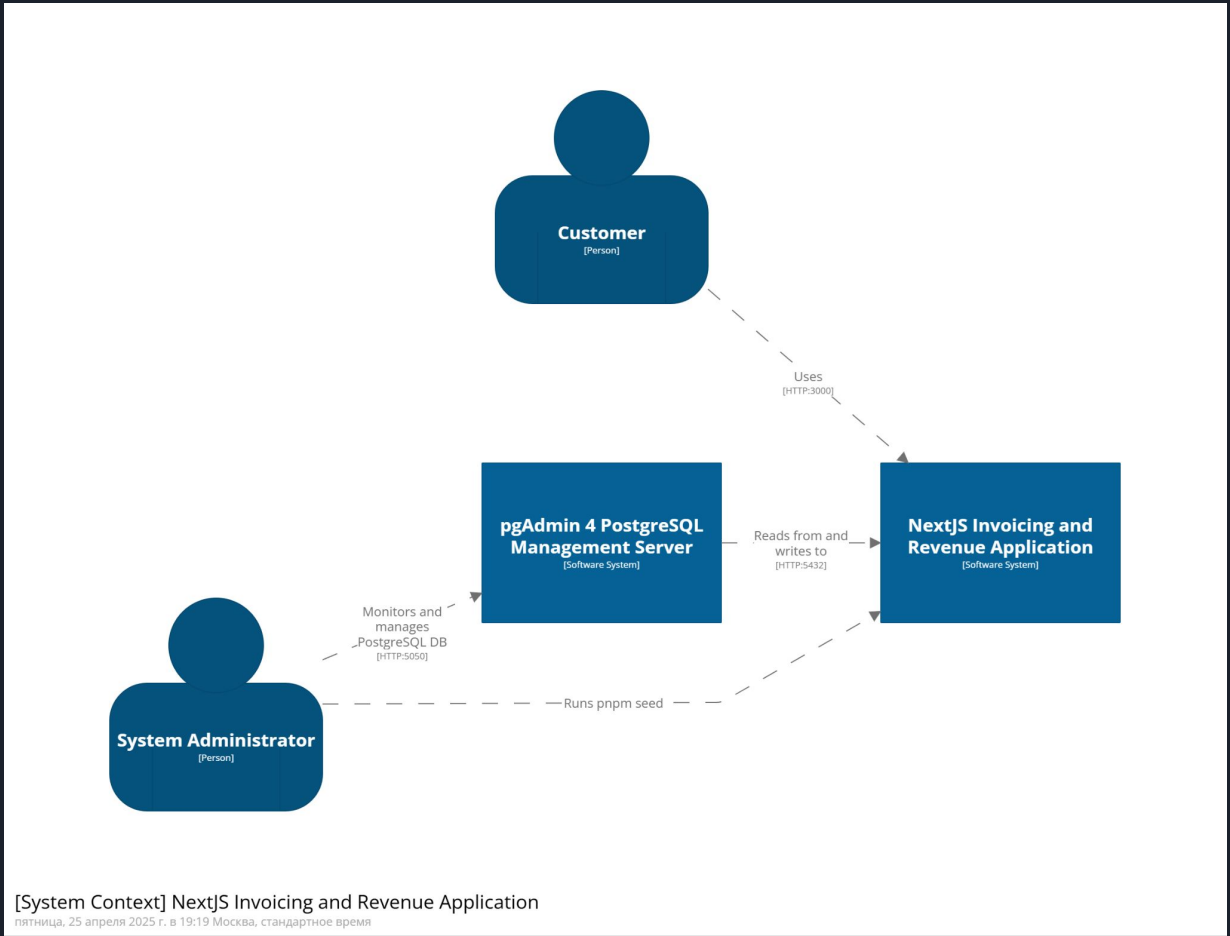
Работаем с контейнерами: pgAdmin (pgadmin_container) (3)

7	Perform configuration & validation	Настройте pgadmin и проверьте данные в БД: Set master password for pgadmin Add new server: test Host: postgres_container Username:postgres Password: changeme Tools / Query Tool select * from book; select * from users; select * from revenue; select * from customers; select * from invoices;

Завершаем работу со стендом NextJS + PostgreSQL+pgAdmin

8	Stop the environment temporarily	Остановите стенд (с возможностью запуска): docker compose stop # start / restart / up allowed [+] Stopping 3/3 ✔ Container pgadmin_container Stopped 29.7s ✔ Container nexxt_container Stopped 26.4s ✔ Container postgres_container Stopped 10.4s
9	Remove the environment	Удалите контейнеры и сети и очистите проект: docker compose down [+] Running 5/5 ✔ Container nexxt_container Removed 0.4s ✔ Container pgadmin_container Removed 0.4s ✔ Container postgres_container Removed 0.2s ✔ Network lesson_008d_backend Removed 1.7s ✔ Network lesson_008d_postgres Removed 1.1s
10	Clear the project	docker image rm diamondondockerhub/nextjsapp:8.1 docker image rm dpage/pgadmin4:9.2 docker image rm postgres:15-alpine sudo rm -rf pgdata pgadmin node_modules # lesson_008d

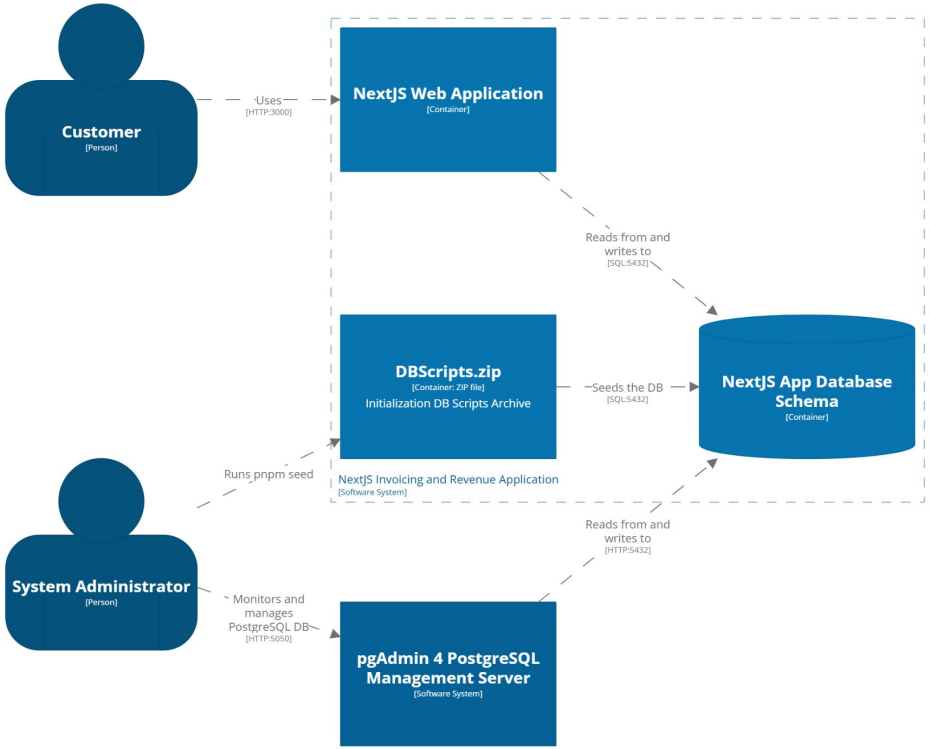
Диаграмма контекста (C1) стенда NextJS + PostgreSQL + pgAdmin



[System Context] NextJS Invoicing and Revenue Application

пятница, 25 апреля 2025 г. в 19:19 Москва, стандартное время

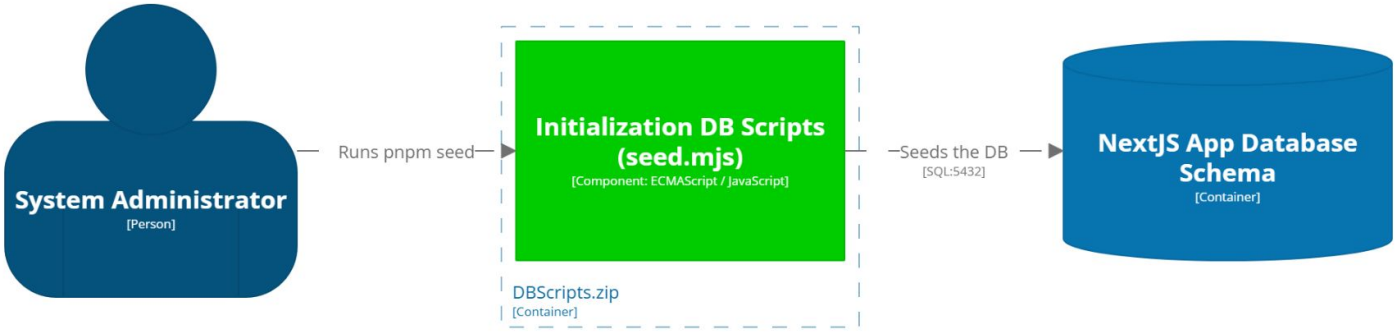
Диаграмма контейнеров (C2) стенда NextJS + PostgreSQL + pgAdmin



[Container] NextJS Invoicing and Revenue Application

пятница, 25 апреля 2025 г. в 19:19 Москва, стандартное время

Диаграмма компонента (C3) стенда NextJS + PostgreSQL + pgAdmin



[Component] NextJS Invoicing and Revenue Application - DBScripts.zip

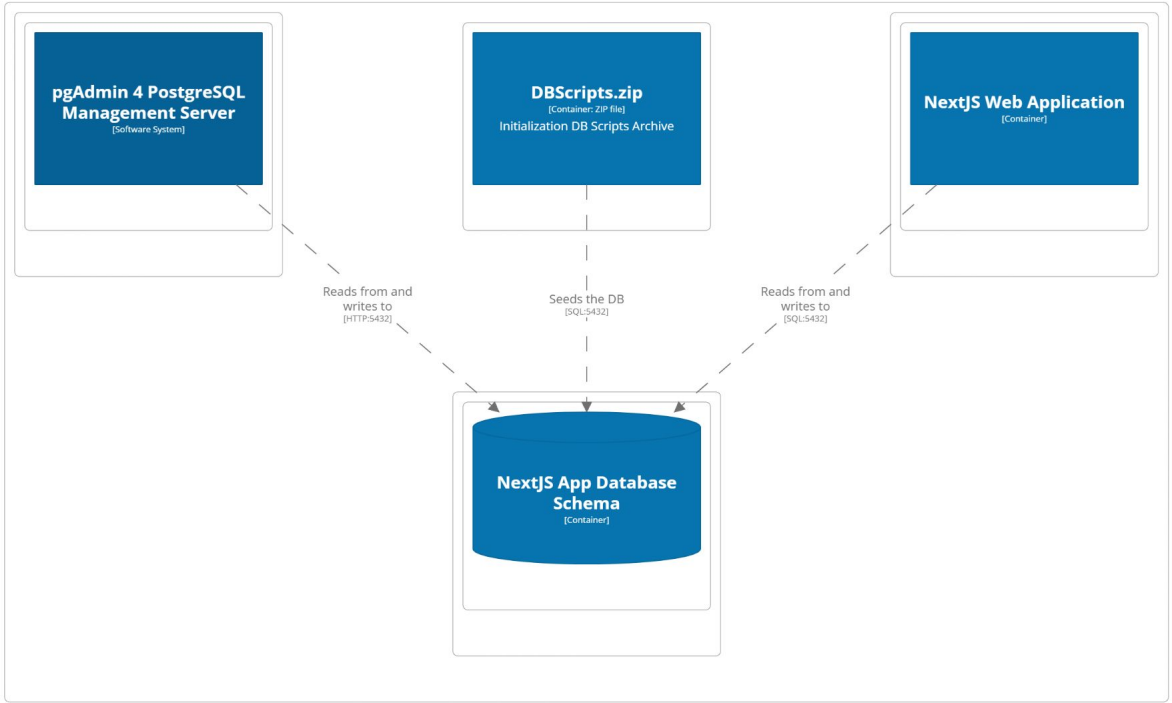
пятница, 25 апреля 2025 г. в 19:19 Москва, стандартное время

NB Обзор модели C4 и structurizr:

rutube: https://rutube.ru/video/819ac2f52c6c0ce7485aac554bad36b1/

youtube: https://www.youtube.com/watch?v=-kKrlFAQfdA

Диаграмма развертывания стенда NextJS + PostgreSQL + pgAdmin



[Deployment] NextJS Invoicing and Revenue Application - Development Environment

An example deployment environment for the NextJS application development

пятница, 25 апреля 2025 г. в 19:19 Москва, стандартное время

Развитие текущей реализации

1	**Dockerfile необходимо оптимизировать по размеру и скорости сборки; при сборке также копировать pnpm-workspace.yaml для безусловной установки пакета bcrypt, необходимого для выполнения pnpm seed во время сборки образа и т.д.**
2	**Инициализационные скрипты выполнять автоматически из папки initdb при запуске контейнера БД, обеспечив при этом хеширование данных**
3	**Обеспечить поддержку HTTPS либо внутри контейнера приложения, либо с помощью отдельного контейнера с webserver’ом nginx как чистого фронтенда**
4	**Защитить доступ к БД: учетные данные должны быть скрыты в разделе “secrets” файла docker-compose.yaml**
5	**Предусмотреть отдельный контейнер для API-сервера приложения**

Подведем итоги

1	**Обсудили 2-контейнерную архитектуру приложения NextJS (Web-приложение и его схема БД)**
2	**Отредактировали dockerfile приложения, настроили и запустили тестовый стенд из трех Docker-контейнеров: NextJS (nexxt_container), PostgreSQL (postgres_container) и pgAdmin (pgadmin_container)**
3	**Прочувствовали ограничения статической отрисовки (static rendering) в смысле невозможности обновления данных из БД после первоначальной сборки**

Следующая лекция -
9. Streaming in NextJS / Потоки данных в NextJS

Презентация доступна для скачивания здесь:

https://dmpsy.club/references/NextJS/lesson_008d_containerization_and_deployment_rus.pdf

Материалы лекции:
https://dmpsy.club/references/NextJS/lesson_008d.tar

Поддержать автора: https://www.donationalerts.com/r/dmitrymak

https://dmpsy.club
postmaster@dmspy.club

https://www.donationalerts.com/r/dmitrymak