Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Пермский государственный аграрно-технологический университет
имени академика Д. Н. Прянишникова»

Кафедра Информационных технологий
и программной инженерии

ОТЧЁТ
О ТЕХНОЛОГИЧЕСКОЙ ПРАКТИКЕ

на тему: «Разработка информационной системы поддержки принятия управленческих решений в системе точного земледелия»

Выполнил:
студент группы ПИм-11
направления подготовки
09.04.03 Прикладная информатика
Нагибин Дмитрий Викторович
Проверил:
доцент кафедры ИТиПИ, к.т.н., доцент
Беляков Андрей Юрьевич

Пермь, 2023 г.

Оглавление

Введение

Ситуация в мире остаётся напряжённой. Аналитики предрекают мировой финансовый кризис [5]. При этом Россия по-прежнему отрезана от мировых рынков, что сказалось, и ещё может сказаться, на уменьшении объёма товаров и технологий, поставляемых из-за рубежа. Нестабильность, с одной стороны, создаёт множество проблем в обыденной жизни, но также несёт и потенциальные перспективы, по выходу из данной ситуации.

В сложившейся ситуации проблема обеспечения населения продуктами сельскохозяйственного производства не перестаёт быть актуальной, а лишь набирает обороты. Отсюда вытекает вывод, что необходимо наращивать производство сельскохозяйственной продукции внутри страны. Наиболее эффективным является интенсивный подход к повышению урожайности, т.к. экстенсивный метод требует увеличения обрабатываемых площадей, что менее затратно, но и менее эффективно. Одним из средств, относящихся к интенсивному подходу, является точное (прецизионное или координатное) земледелие.

Целью технологической практики является апробация технологий для развёртывания имеющихся наработок на серверной машине с применением технологий виртуализации и контейнеризации, а также с возможностью подключения к системе по протоколу VPN.

Среди задач стоит выделить:

— провести обзор технологий виртуализации;

— провести обзор технологий контейнеризации;

— провести обзор актуальных протоколов VPN;

— развернуть имеющиеся наработки с применением перечисленных технологий на серверной машине.

1. Постановка проблемы исследования

Анализ имеющихся решений в сфере точного земледелия показал, что многие из имеющихся на рынке систем поддержки принятия управленческих решений в сфере точного земледелия полагаются на зарубежные технологии и инвестиции. Отсюда вытекает актуальность работы в данном направлении при сложившихся условиях рынка. Также в своём большинстве конкурентные системы интегрируются в работу агропредприятия в целом и не делают упор на проблемы точного земледелия [3].

Разработка системы поддержки принятия решений в сфере точного земледелия представляет из себя комплексный процесс, ключевой частью которого является проектирование модулей и подсистем, разрабатываемой информационной системы.

Обзор имеющихся аналогов привёл к формированию списка наиболее необходимых функций для такой системы [3]. Ключевые возможности разрабатываемой системы представлены на рисунке 1.1.

Рисунок 1.1— Наиболее важные особенности ИС точного земледелия

2. Анализ технологий проектирования

В системе поддержки принятия решений в сфере точного земледелия будет храниться большой объём данных, поскольку для более точного прогнозирования, например урожайности, потребуются данные за предыдущие, до запуска системы, года. Это позволит намного точнее рассчитывать прогнозы с учётом всех особенностей местного климата, в частности Пермского края. К тому же обновление информации в системе должно быть в режиме реального времени, а это значит, что и данные со всех датчиков будут поступать в режиме реального времени и эти данные тоже будет необходимо сохранять в базе данных для дальнейшего использования. Естественно, данные необходимо будет группировать и минимизировать, в целях более экономного использования дискового пространства.

Поскольку система должна быть многопользовательской, то наиболее оптимальным вариантом будет «облачный» сервис. Такой сервис будет построен по клиент-серверной архитектуре, что позволит сделать систему кроссплатформенной изначально. Пользователи будут подключаться к серверу через веб-приложение в обычном веб-браузере, который одинаково хорошо позволит использовать разрабатываемую ИС как на системах семейства Windows, так и на Linux, MacOS и Android.

В качестве веб-сервера приложений стоит рассмотреть Django. В отличие от ASP.NET, данное решение не завязано на экосистеме конкретной компании, в частности Microsoft. Также Django, в отличие от решений на базе Node.js или PHP, является комплексным фреймворком, включающем в себя множество функций, например: механизм авторизаций в Django уже встроен и нет необходимости прибегать к применению сторонних библиотек, к тому же данное решение имеет модульную архитектуру, что в дальнейшем упростит реализацию микросервисной архитектуры. Стоит отметить и работу с базой данных в этом фреймворке, в данном случае предполагается использование PostreSQL. Связь Django с базой данных осуществляется по технологии ORM (Object–Relational Mapping).

Для разработки клиентской части выбран фреймворк Vue.js. Данное решение является довольно популярным при разработке, имеет широкий функционал и не несёт лишних сложностей при разработке. Vue.js позволяет создавать комплексное веб-приложение с довольно обширным функционалом, например, не потребуется обновления всей страницы целиком для обновления каких-либо данных на этой странице. В случае необходимости будет обновлены лишь требуемые компоненты веб-страницы. Также для данного решения имеется множество плагинов, расширяющих функционал.

Взаимодействие клиентов с сервером будет осуществляться по RESTAPI. Это позволит выделить клиентскую часть ИС в отдельное приложение. В дальнейшем при помощи RESTAPI можно будет подключать клиентов вне зависимости от типа клиентского приложения, т.е. один и тот же сервер будет схожим образом «общаться» как с веб-клиентом, так и с клиентом-мобильном приложением, так и с нативными клиентом в какой-либо операционной системе.

На рисунке 2.1 представлена приблизительная схема взаимодействия компонентов системы поддержки принятия решений в сфере точного земледелия. Для более удобного развёртывания каждого из компонентов системы стоит использовать средства контейнеризации. Использование контейнеров позволит относительно легко переносить уже полностью настроенные и готовые к эксплуатации компоненты между серверными машинами. Практически каждый компонент, из изображённых, будет работать внутри контейнера.

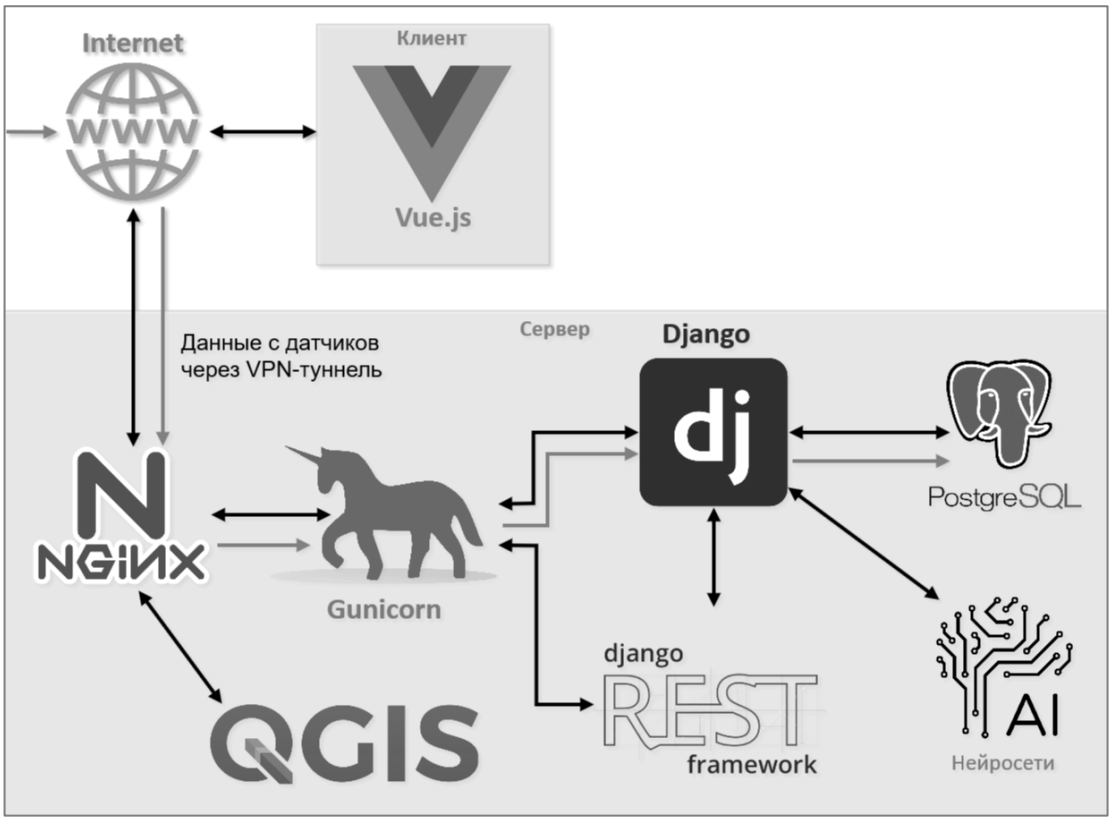

Рисунок 2.1 — Предполагаемая схема реализации

Также на схеме показаны нейросети и геоинформационная система. Для реализации поставленных задач будут использоваться два типа нейронных сетей: свёрточные и рекуррентные нейронные сети. В качестве ГИС будет использоваться система с открытым исходным кодом QGIS.

Поскольку наработки по данному проекту уже имеются и, раннее они были размещены на удалённом сервере, а теперь будут размещены на локальном сервере, используемом и для других задач, то в рамках данной практики эти наработки будет развёрнуты при помощи системы виртуализации.

Среди наиболее известных систем виртуализации можно выделить:

— VMware ESXi;

— Microsoft Hyper-V;

— Open Virtualization Alliance KVM;

— Oracle VM VirtualBox.

Они все достаточно универсальны, однако, у каждого из них имеются определенные особенности, которые следует всегда учитывать на этапе

выбора: стоимость развёртывания/обслуживания и технические характеристики. Стоимость коммерческих лицензий VMware и Hyper-V весьма высока. KVM же напротив, полностью бесплатен и достаточно прост в работе, особенно в составе готового решения на базе Debian Linux под названием Proxmox Virtual Environment [2].

Чтобы облегчить работу с каждым сервисом по отдельности, каждый из сервисов стоит разместить в программных контейнерах, это своего рода тоже виртуальные машины, но требующие намного меньше вычислительных ресурсов для своего функционирования. Контейнеры изолируют запущенные внутри них сервисы от хостовой машины (самого сервера), при этом имеют гибкие настройки. Контейнер, собранный на одном ПК, может быть с лёгкостью запущен на другом ПК, поскольку все настройки над сервисом проводятся автоматически системой контейнеризации. В качестве вариантов систем контейнеризации рассматривались Docker и Kubernetes.

Docker контролирует приложения и во время разработки, и в во время работы, помогает управлять хранилищем, памятью и правами приложений, обеспечивает согласованную среду на любом совместимом хосте (*nix или Windows). «Библиотека» уже готовых к использованию контейнеров, Docker Hub, представлен на рисунке 2.2.

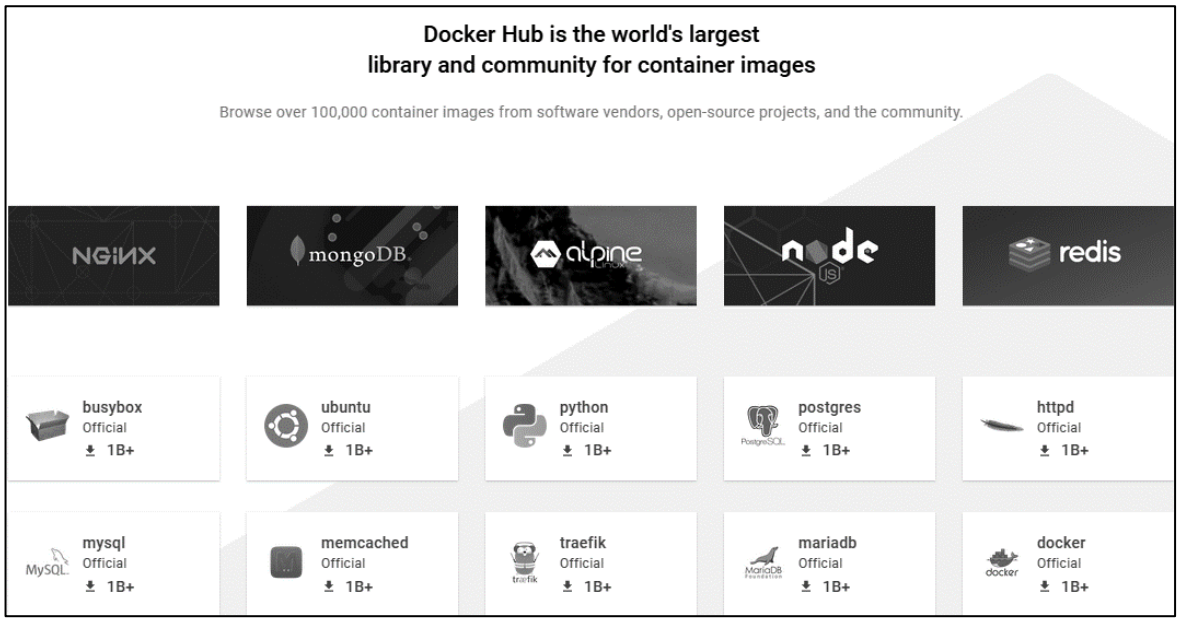

Рисунок 2.2 — Веб-страница Docker Hub

Среди особенностей Docker выделяют:

— простая и быстрая настройка;

— высокая производительность;

— изоляция – Docker использует для запуска приложений контейнеры;

— качественное управление безопасностью.

Kubernetes – инструмент управления контейнерами, автоматизирующий развёртывание. Это платформа с открытым исходным кодом, разработанная компанией Google, а теперь управляемая Cloud Native. Kubernetes помогает в обновлении приложений простым и быстрым способом, управляет рабочей нагрузкой и планированием контейнеров в кластере, автоматизирует многие ручные процессы, например, управлением приложениями в контейнере и их масштабированием.

Особенности Kubernetes:

— автоматизация ручных процессов;

— балансировка нагрузки (Kubernetes распределяет сетевой трафик и поддерживает стабильность развёртывания);

— самовосстановление (инструмент перезапускает отказавшие контейнеры, перемещает, а также «убивает» контейнер, не отвечающий шаблону пользователя);

— инструментирование хранилища (пользователи могут автоматически монтировать систему хранения на свой выбор).

Поскольку в разрабатываемой ИС предполагается создание микросервисной архитектуры, был выбран Docker. Также Docker не требует слишком сложных настроек и нетребователен в эксплуатации, по сравнению с Kubernetes, который больше подходит для крупных компаний с большим штатом разработчиков [1].

Стоит также отметить, что получение данных, например от метеодатчиков, будет происходить не непосредственно от самих датчиков, а от устройств-хабов, принимающих показания датчиков, а затем, «пачкой» отправляющих эти показания на сервер для дальнейшей обработки. Поэтому

необходимо рассмотреть возможность реализации надёжной и защищённой от вмешательства связи между хабом и сервером, поскольку они будут соединены через интернет. В качестве механизма защиты рассматривается настройка VPN-туннелей между сервером и устройствами-хабами. Для реализации были рассмотрены два наиболее актуальных протоколs для организации связи между сервером и хабом: OpenVPN и WireGuard.

Архитектурно WireGuard более безопасен за счёт того, что поверхность атаки значительно меньше по сравнению с OpenVPN. Тем не менее, OpenVPN считается очень безопасным, надёжным и конфиденциальным, многократно пройдя независимый аудит кода, а также довольно гибким в настройке. За счёт этого OpenVPN выигрывает в плане безопасности.

Однако синтетические замеры скорости соединения внутри VPN-туннеля позволяют предположить, что WireGuard быстрее, чем OpenVPN. К тому WireGuard, за счёт меньших мер по защиты VPN-соединения, требует намного меньших аппаратных ресурсов [6]. Поэтому выбор остановился на WireGuard.

3. Реализация функционала информационной системы

В первую очередь, для развёртывания всех остальных сервисов, стоит произвести установку и настройку среды виртуализации Proxmox. Proxmox поставляется в виде ISO-образа ОС для установки непосредственно на оборудование, выполняющее роль сервера — рисунок 3.1.

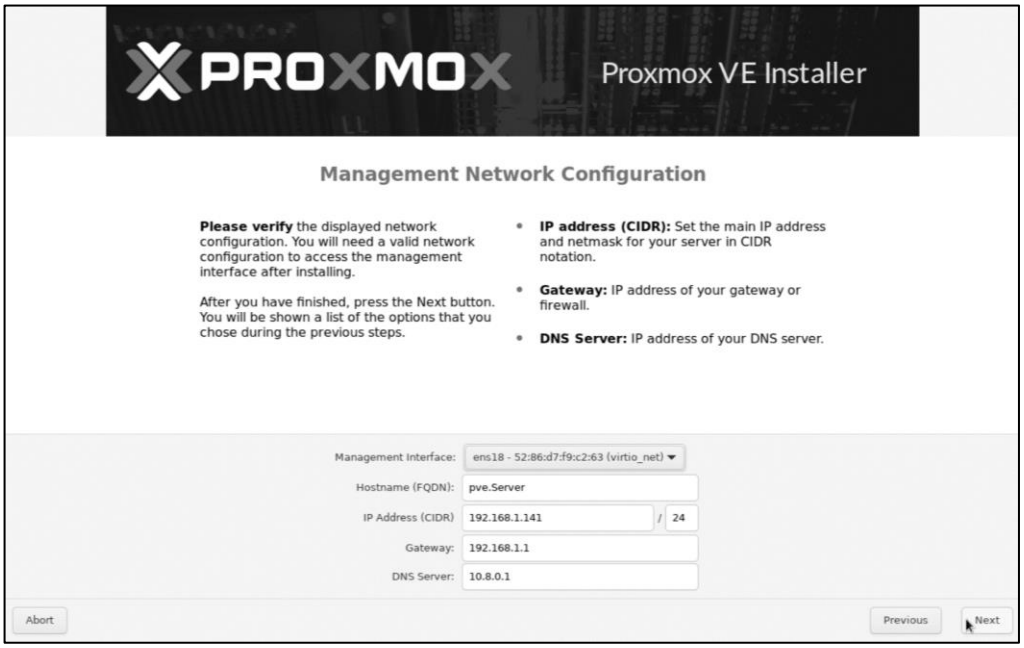

Рисунок 3.1 — Установка Proxmox

Работа по настройке виртуальных машин выполняется в веб-интерфейсе Proxmox с любого устройства в одной с сервером локальной сети. Также работа с системой возможна и на самом сервере, но только через консольный интерфейс, поскольку оболочка рабочего стола не предусмотрена для систем такого рода. То, как выглядит веб-интерфейс уже настроенной среды виртуализации Proxmox, представлен на рисунке 3.2.

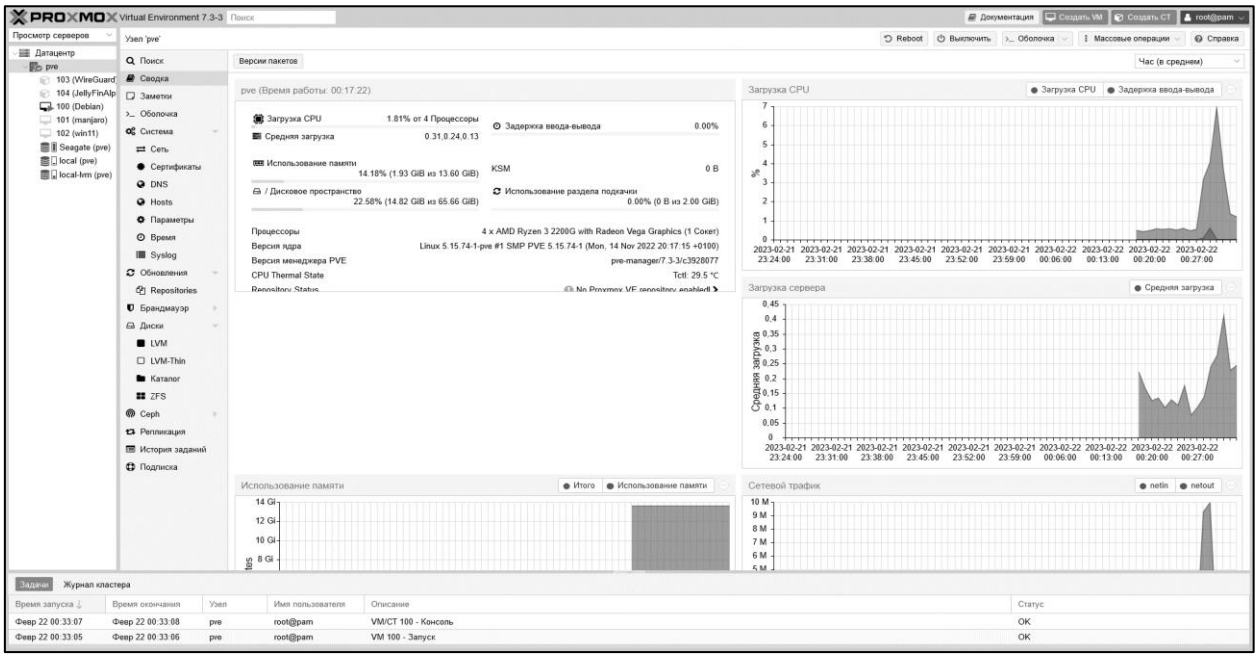

Рисунок 3.2 — Веб-интерфейс Proxmox

В среде Proxmox можно создавать не только виртуальные машины, но и Linux-контейнеры. В рамках данного проекта была создана виртуальная машина на базе Debian Linux. Взаимодействие с виртуальной машиной осуществляется либо в веб-интерфейсе через консоль, либо через SSH-подключение. Например, на рисунке Рисунок 3.2 представлена консоль созданной виртуальной машины. Прочие настройки виртуальной машины можно проводить как в веб-интерфейсе, так и через консоль среды Proxmox.

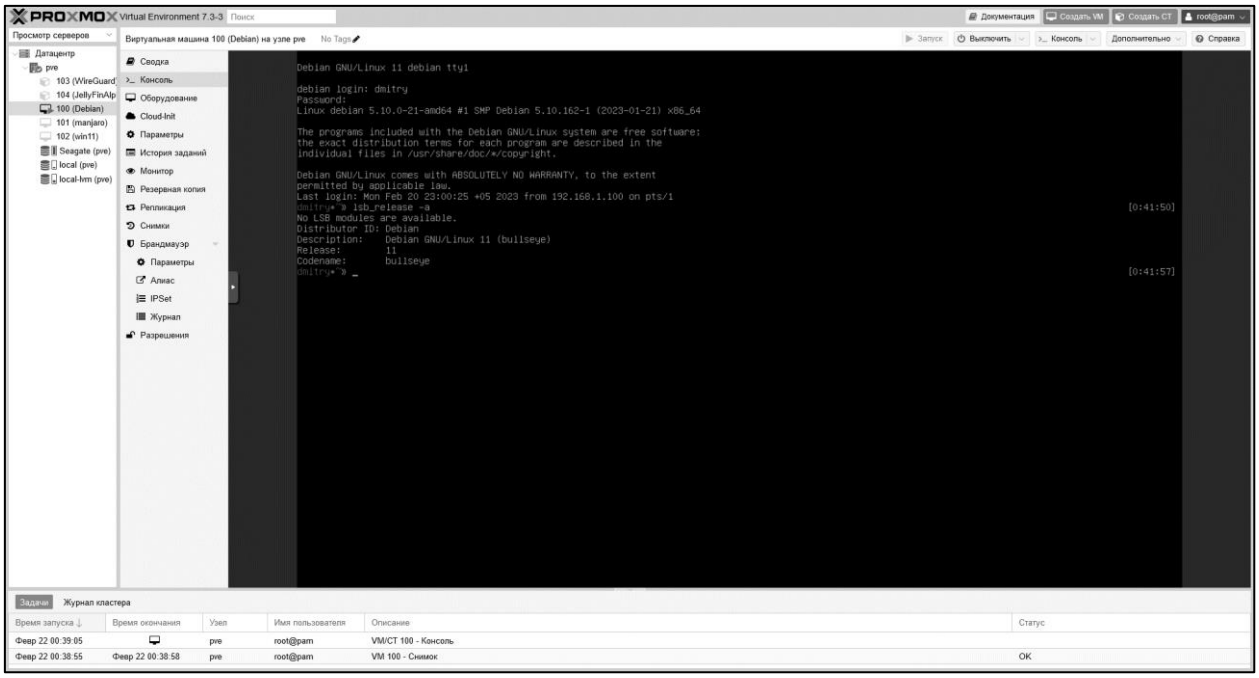

Рисунок 3.3 — Взаимодействие с виртуальной машиной через консоль

Следующие шаги по настройке уже производились внутри виртуальной машины. В рамках данного отчёта не будет описания всех команд, вводимых в консоль виртуальной машины, а лишь результаты их выполнения.

После установки ряда утилитарных приложений в виртуальную машины через пакетный менеджер, встроенный в Debian, под названием «Aptitude», была выполнена установка среды контейнеризации Docker. В первую очередь, для более удобной работы с контейнерами был скачан с Docker Hub и установлен инструмент для управления контейнерами «Portainer». Данный инструмент сам по себе поставляется в виде контейнера и позволяет манипулировать контейнерами и всей сопутствующей инфраструктурой через веб-интерфейс, представленный на рисунке 3.4.

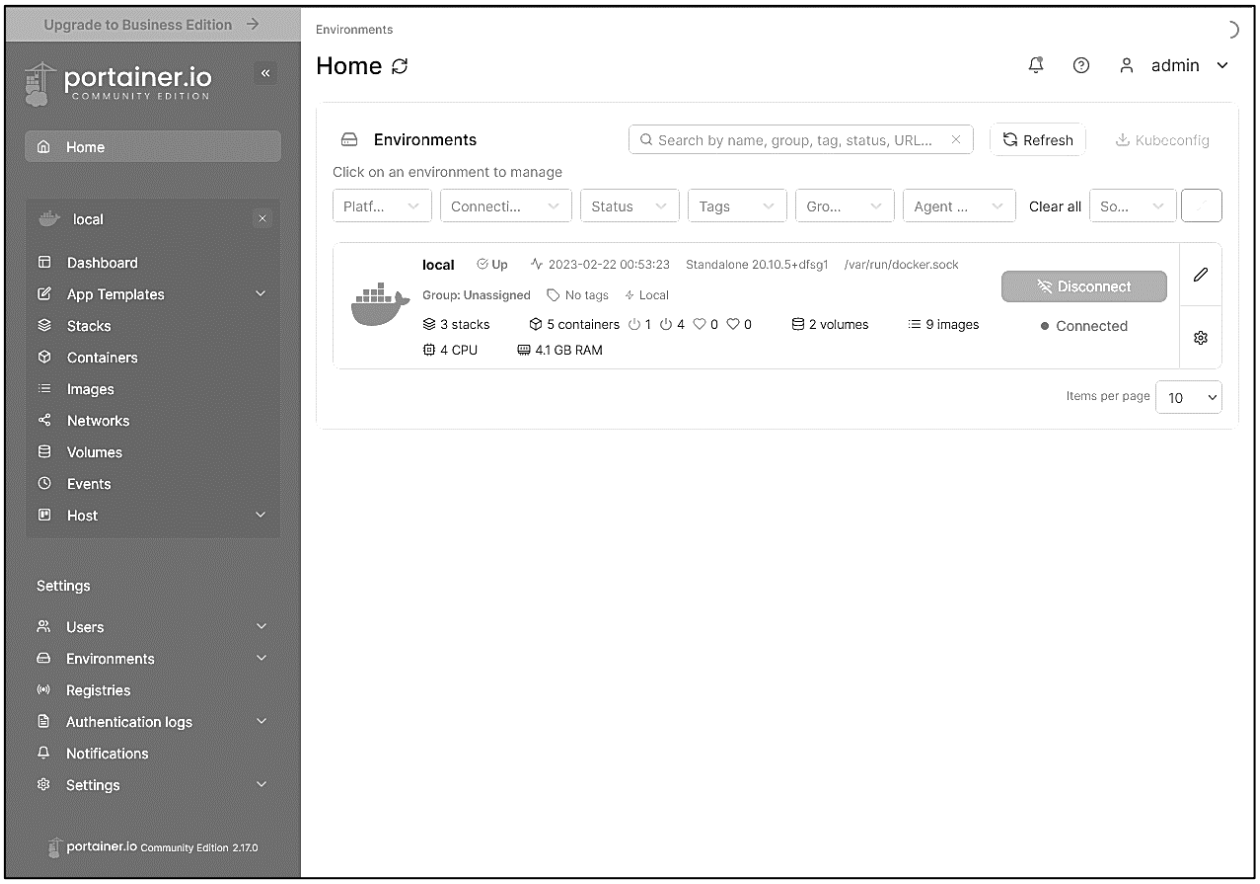

Рисунок 3.4 — Веб-интерфейс инструмента манипулирования контейнерами «Portainer»

На рисунке 3.5 представлен пример настройки контейнера с Django в веб-интерфейсе Portainer.

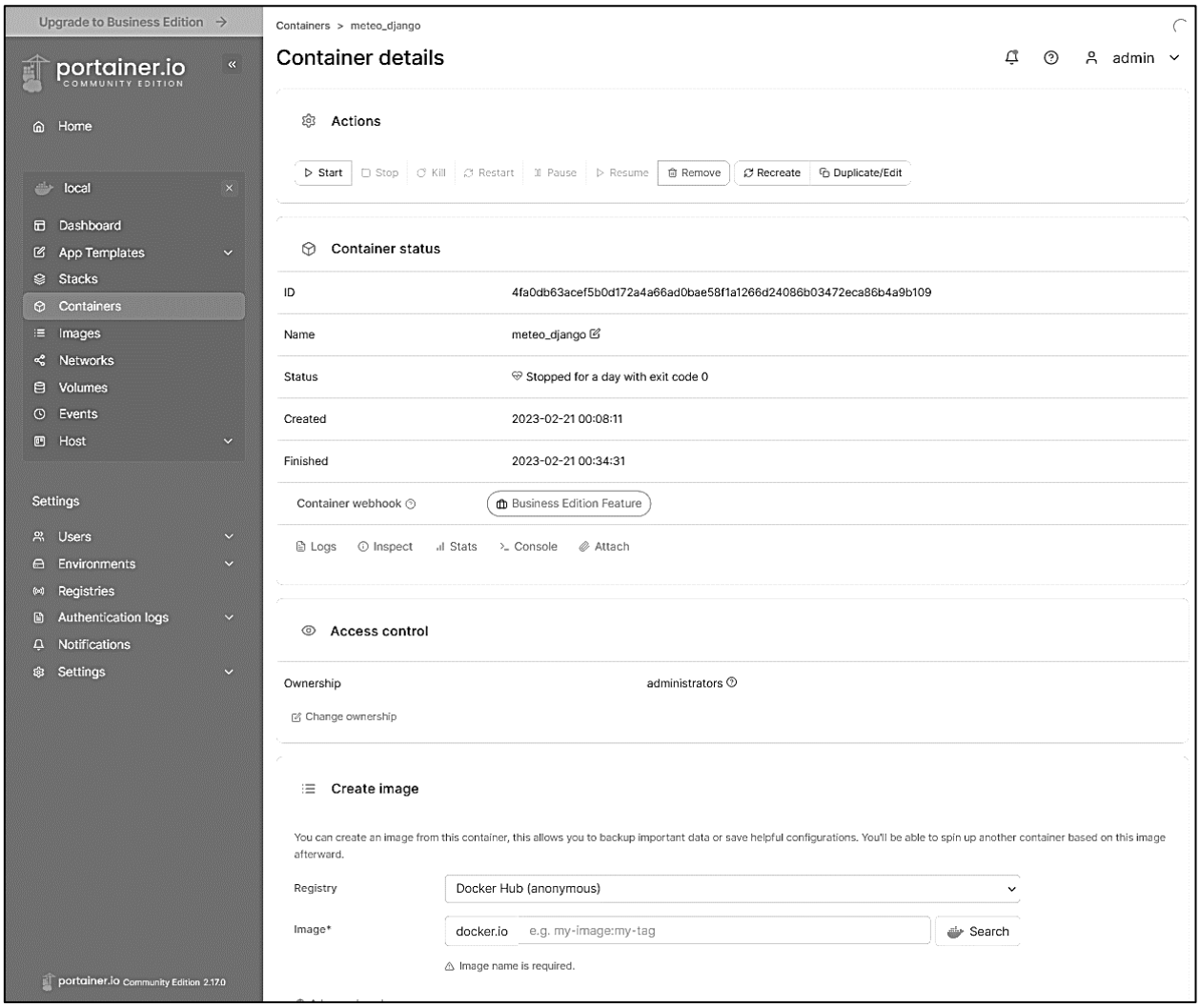

Рисунок 3.5 — Пример настроек определённого контейнера

Теперь можно приступать к созданию Docker-контейнеров. В рамках данной практики будут созданы контейнеры для сервера приложений Django, системы управления реляционными базами данных PostreSQL, — это всё можно отнести к серверной части разрабатываемой ИС (Backend), и Vue-приложения, совмещённого с Nginx в рамках одного контейнера (для полноценной работы Vue-приложения требуется веб-сервер), — клиентская часть разрабатываемой ИС (Frontend). Также в отдельный контейнер будет вынесен веб-сервер Nginx, общий в данном случае для обеих частей разрабатываемой ИС.

Создание контейнеров производилось при помощи встроенного в Docker инструмента Docker-compose, который представляет собой автоматическую систему сборки контейнеров по описанию из файла «YML» формата, написанного разработчиком. На рисунке 3.6 представлен файл docker-

compose, согласно которому Docker производит настройку Django, PostreSQL и Nginx.

```yaml
version: "3.7"
services:

  django:
    build:
      dockerfile: ./app/Dockerfile
      context: .
    container_name: meteo_django
    env_file:
      - ./env/.env
    command: gunicorn meteogis.wsgi:application --bind 0.0.0.0:8000
    expose:
      - 8000
    depends_on:
      - postgresql

  postgresql:
    image: postgres:alpine
    container_name: meteo_postgresql
    volumes:
      - postgres_data:/var/lib/postgresql/data/
    environment:
      - POSTGRES_USER=psql_user
      - POSTGRES_PASSWORD=04post@sql06
      - POSTGRES_DB=meteo_db

  nginx:
    build: ./nginx
    container_name: meteo_nginx
    volumes:
      - /home/dmitry/project/meteo_back/app/static:/home/app/web/static
    ports:
      - "80:80"
    depends_on:
      - django

volumes:
  postgres_data:
```

Рисунок 3.6 — Листинг всей бекэнд-части ИС, файл «docker-compose.yml»

Сборка каждого сервиса производится согласно описанию из этого файла. Например, для PostreSQL используется готовый контейнер («image»), а для Django и Nginx предусмотрена индивидуальная настройка контейнеров, описанная в файлах Docker («build» либо «dockerfile»). Также для контейнеров можно указывать, если необходимо, имена («container_name»), проброс портов внутрь контейнера, например, для доступа извне («ports» и «expose»), переменные окружения («environment» и «env_file»), внешние хранилища («volumes»), а также зависимости одних контейнеров от других («depends_on»). Также могут быть описаны команды консоли, выполняемые в конце запуска контейнера («command»).

На рисунке 3.7 представлена структура файлов и папок внутри проекта до третьего уровня вложенности.

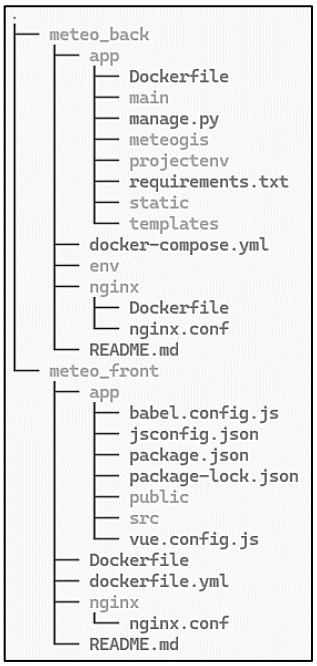

Рисунок 3.7 — Структура фалов и директорий проекта

На рисунке 3.8 представлен Dockerfile по созданию контейнера для Django. Для создания контейнера используется облегчённый alpine-образ контейнера для языка Python (строка 1). Также в данном файле устанавливается ряд переменных, применяемых для конкретного контейнера (строки «ENV»), и создаётся рабочая директория для Django внутри контейнера («WORKDIR»), куда все файлы и копируются («COPY»). Затем запускается установка необходимых для Django зависимостей («RUN»).

```
1  FROM python:alpine
2
3  ENV PYTHONDONTWRITEBYTECODE 1
4  ENV PYTHONUNBUFFERED 1
5
6  WORKDIR /usr/src/project
7
8  COPY ./app/requirements.txt /usr/src/requirements.txt
9
10 RUN apk add python3-dev musl-dev
11
12 RUN pip3 install --upgrade pip
13 RUN pip3 install -r /usr/src/requirements.txt
14 COPY ./app /usr/src/project
```

Рисунок 3.8 — Листинг Dockerfile-файла сборки сервера-приложений Django

Для сборки контейнера для веб-сервера Nginx используется свой Dockerfile, представленный на рисунке 3.9. Здесь происходят схожие процессы, что и при построении контейнера для Django.

```
FROM nginx:alpine

RUN rm /etc/nginx/conf.d/default.conf
COPY nginx.conf /etc/nginx/conf.d

ENV APP_HOME=/home/app/web
RUN mkdir -p $APP_HOME
RUN mkdir -p $APP_HOME/static
WORKDIR $APP_HOME
```

Рисунок 3.9 — Листинг Dockerfile-файла сборки веб-сервера Nginx

Файл, копируемый внутрь контейнера Nginx, представлен на рисунке 3.10. Данный файл является файлом конфигурации веб-сервера Nginx, в котором описано, что Nginx должен принимать подключения на порту 80 и перенаправлять их на сервер-приложений Django, расположенный на 8000 порту, также в этом файле указано местоположение статических файлов, которыми могут являться, например, CSS-стили, применяемые на веб-страницах разрабатываемой системы.

```
upstream meteogis {
    server django:8000;
}

server {

    listen 80;

    location / {
        proxy_pass http://meteogis;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header Host $host;
    }

    location /static/ {
        alias /home/app/web/static/;
    }
}
```

Рисунок 3.10 — Листинг файла конфигурации веб-сервера «nginx.conf»

Конфигурация контейнера для Vue на данный момент производится отдельно, файл которой представлен на рисунке 3.11. Функционал данного файла «docker-compose.yml» аналогичен, описанному выше.

```
version: '3.7'
services:

  vue:
    container_name: vue
    build:
      context: .
      dockerfile: Dockerfile
    ports:
      - '80:80'
```

Рисунок 3.11 — Листинг файла «docker-compose.yml» фронтэнд-части ИС

Но основная работа по созданию контейнера представлена на рисунке 3.12. В данном файле описывается создание контейнера на основе

NodeJS, поскольку он необходим для работы Vue. В процессе сборки контейнера происходит установка всех зависимостей, необходимых для полноценной работы Vue-приложения. Затем запускается процесс сборки Vue-приложения из исходников (строка 8).

```
1  FROM node:alpine  as build
2  WORKDIR /app
3  ENV PATH /app/node_modules/.bin:$PATH
4  COPY app/package.json /app/package.json
5  RUN npm install --silent
6  RUN npm install @vue/cli@3.7.0 -g
7  COPY app /app
8  RUN npm run build
9
10 # production environment
11 FROM nginx:alpine
12 COPY --from=build /app/dist /usr/share/nginx/html
13 RUN rm /etc/nginx/conf.d/default.conf
14 COPY nginx/nginx.conf /etc/nginx/conf.d
15 EXPOSE 80
16 CMD ["nginx", "-g", "daemon off;"]
```

Рисунок 3.12 — Листинг Dockerfile-файла сборки Vue с Nginx

После сборки Vue-приложения запускается установка Nginx, процесс которой идентичен описанной выше, за исключением того, что в данном случае Nginx будет обслуживать запросы, адресованные лишь Vue-приложению, а не ИС в целом.

В заключение, будет описана настройка VPN-соединения. На данный момент практическая польза VPN-соединения незначительна, так как ещё не создана инфраструктура для работы всей ИС. Поэтому VPN-сервер был развёрнут на удалённом VPS-сервере. Приблизительная конфигурация WireGuard-сервера представлена ниже, на рисунке 3.13. А на рисунке 3.14 представлена конфигурация WireGuard-клиента.

```
1  [Interface]
2  Address = 10.0.0.1/24
3  Address = fd40:40:40::1/64
4  SaveConfig = true
5  PostUp = iptables -A FORWARD -i %i -j ACCEPT; iptables -A FORWARD -o %i -j ACCEPT;
   iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
6  PostDown = iptables -D FORWARD -i %i -j ACCEPT; iptables -D FORWARD -o %i -j ACCEPT;
   iptables -t nat -D POSTROUTING -o eth0 -j MASQUERADE
7  ListenPort = 12345
8  PrivateKey = +I+h1UKC1Y5j+jbLuC
9
10 [Peer]
11 PublicKey = AIH6oBH8ZEXAIUZx
12 PresharedKey = t+viqP+I7kqxR
13 AllowedIPs = 10.0.0.10/32, 192.168.1.0/24, fd40:40:40::10/128
14 Endpoint = 9.8.7.6:98765
15 PersistentKeepalive = 25
```

Рисунок 3.13 — Настройки WireGuard сервера

Настройка по протоколу WireGuard намного проще таковой, например, в протоколе OpenVPN. Серверная и клиентская конфигурации во многом схожи, за исключением некоторых параметров.

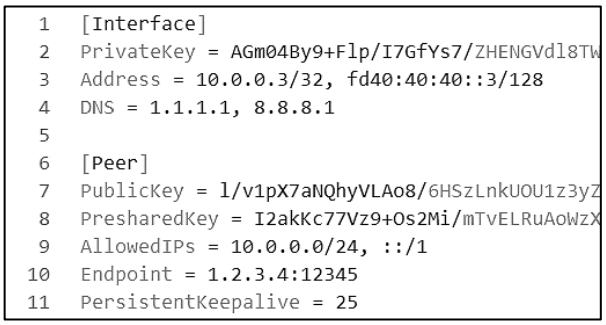

Рисунок 3.14 — Настройки WireGuard клиента

Обе конфигурации содержат два основных блока: блок «Interface» и блок «Peer».

В первом блоке описываются настройки сетевого адаптера текущей стороны (сервер/клиент). Тут же описывается приватный ключ (уникальный идентификатор данной точки подключения) — в случае и сервера и клиента. Также описывает IP-адресация внутри VPN-туннеля и настройки DNS, распространяемые также лишь внутри VPN-туннеля. Для сервера прописывается порт, на котором WireGuard будет ожидать подключения от клиентов, а также параметры «PostUp» и «PostDown». Первый создаёт правила маршрутизации для брандмауэра при установлении соединения, а второй удаляет данные правила. Причём точно такая же настройка с правилами брандмауэра может проводиться и на клиентской стороне.

Во втором блоке, «Peer», описывается место назначение подключения, приватный и «Preshared» ключи, разрешённые IP-адреса, благодаря чему автоматически создаются статические маршруты между участниками VPN-соединения и пакеты данных «знают куда идти» и могут проходить от одного клиента к другому по VPN-туннелю через WireGuard-сервер. Также тут прописывается IP-адрес конечного подключения, например, для клиента прописывается IP-адрес WireGuard-сервера, также тут может быть прописана периодичность отправки пакетов для поддержания VPN-соединения в активном состоянии.

Заключение

В разработке данной системы в основном заинтересованы сельхозтоваропроизводители, поскольку система позволит предотвратить развитие сорных растений и болезней, а также менее затратно оптимизировать операционные расходы и повысить урожайность в среднем на 15-20%, поскольку точные и оперативные данные позволяют вовремя реагировать на постоянно меняющиеся условия среды [7].

При проектировании такой масштабной информационной системы стоит учитывать множество факторов. Часть из них была пересмотрена входе работы над данным отчётом. Например, необходимо практически полностью переработать фронтэнд часть ИС, поскольку на данный момент текущая реализация не позволяет отделить Vue от Django в отдельные по функционалу модули. Но перед этим придётся переработать и бекэнд часть, потому что без нормально функционирующей серверной части не сможет нормально работать клиентская часть разрабатываемой ИС.

Практика использования Docker-контейнеров показала себя с наилучшей стороны, стоит использовать такой подход и в дальнейшем. Вполне возможно, что в ходе реализации придётся пересмотреть ещё большую часть аспектов системы и от некоторых из них отказаться в пользу более подходящих по тем или иным причинам.

Список источников

1. В чём разница между Docker и Kubernetes? [Электронный ресурс]. URL: https://proglib.io/p/v-chem-raznica-mezhdu-docker-i-kubernetes-2020-06-02 (Дата обращения: 18.02.2023).

2. Магия виртуализации: вводный курс в Proxmox VE [Электронный ресурс]. URL: https://habr.com/ru/company/selectel/blog/483236/ (Дата обращения: 17.02.2023).

3. Нагибин Д.В. Применение информационных технологий для решения задач в системе точного земледелия // проблемы и перспективы развития АПК региона. материалы краевой студенческой научно-практической конференции. Пермь: ИПЦ Прокростъ, 2023. С. 458.

4. Сельское хозяйство будущего: нейронные сети научились предсказывать динамику роста растений [Электронный ресурс]. URL: https://naked-science.ru/article/column/selskoe-hozyajstvo-budushhego-nejronnye-seti-nauchilis-predskazyvat-dinamiku-rosta-rastenij (Дата обращения: 10.02.2023).

5. ЦБ описал сценарий глобального кризиса в 2023-м. Какие рынки под угрозой? [Электронный ресурс]. URL: https://quote.rbc.ru/news/article/613735ac9a7947010bee0e51 (Дата обращения: 15.02.2023).

6. Что лучше выбрать: Wireguard или OpenVPN? Любимый VPN Линуса Торвальдса [Электронный ресурс]. URL: https://habr.com/ru/company/ruvds/blog/537010/ (Дата обращения: 10.02.2023).

7. AgroTech: как фермеров пытаются подружить с искусственным интеллектом [Электронный ресурс]. URL: https://trends.rbc.ru/trends/industry/614b6fd09a79470280d775ea (Дата обращения: 10.02.2023).