



Министерство науки и высшего образования Российской Федерации  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«Московский государственный технический университет  
имени Н.Э. Баумана  
(национальный исследовательский университет)»  
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

## ОТЧЁТ ПО ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ

Студент Романов Семен Константинович

Группа ИУ7-65Б

Тип практики Производственная

Название предприятия ООО «СитиСофт»

Студент

подпись, дата

Романов С. К.

фамилия, и.о.

Руководитель практики

(от университета)

подпись, дата

Толпинская Н. Б.

фамилия, и.о.

Руководитель практики

(от предприятия)

подпись, дата

Батин Р. Е.

фамилия, и.о.

Оценка

2023 г.

## Содержание

<b>ВВЕДЕНИЕ</b>	<b>3</b>
<b>1 Характеристика предприятия</b>	<b>4</b>
<b>2 Характеристика проекта для производственной практики</b>	<b>5</b>
<b>3 Характеристика индивидуального задания для производственной практики</b>	<b>6</b>
<b>4 Описание выполнения задания</b>	<b>6</b>
4.1 Командная разработка . . . . .	6
4.2 Анализ используемых технологий . . . . .	7
4.2.1 Rust . . . . .	7
4.2.2 OpenAPI . . . . .	8
4.2.3 Axum . . . . .	8
4.2.4 TypeScript . . . . .	8
4.2.5 Next.js . . . . .	9
4.2.6 React . . . . .	9
4.3 Разработка серверной части приложения . . . . .	9
4.4 OpenAPI Спецификация . . . . .	9
4.4.1 Клиентская часть . . . . .	13
4.4.2 Методы авторизации . . . . .	15
4.4.3 Методы публичного API . . . . .	16
4.5 Демонстрация работы веб-приложения . . . . .	17
<b>ЗАКЛЮЧЕНИЕ</b>	<b>22</b>
<b>СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ</b>	<b>23</b>

## ВВЕДЕНИЕ

BOB — это распределённая система хранения данных типа ключ-значение [1]. BOB проектировался для быстрой и надёжной работы с бинарными данными средних размеров (например, с изображениями). BOB постоянно развивается и обладает множеством запланированных задач разной сложности.

Цель производственной практики — овладеть навыками разработки систем хранения данных, Web-приложений [2], open source-проектов [3]

Задачи проекта:

Разработать Web-приложение [2], включающее в себя:

- 1) страницу авторизации;
- 2) домашнюю страницу с основными параметрами кластера;
- 3) страницу с нодами;
- 4) страницу с подробной информацией о ноде;
- 5) страницу с раскладкой виртуальных дисков.

Индивидуальные задачи:

- 1) получить навыки командной работы;
- 2) спроектировать и разработать клиент для взаимодействия с API [4] Боба;
- 3) спроектировать собственный API [4];
- 4) разработать серверную часть веб-приложения, интегрируя в него ранее разработанный клиент и API [4];
- 5) провести тестирование итогового Web-приложения.

## **1 Характеристика предприятия**

Компания «СитиСофт» предоставляет высококачественные услуги для тех, кто стремится улучшить свой бизнес путем внедрения или обновления программно - аппаратной инфраструктуры.

Продукты «СитиСофт» — это специальные технические средства и комплексы, серверы и компьютерные компоненты, активное и пассивное сетевое оборудование, Системы Хранения Данных и специализированное ПО, а также решения на базе оборудования ведущих мировых производителей и многое другое.

Предлагаемые услуги включает самые востребованные на сегодняшний день IT-проекты, позволяющие масштабировать решения, получить максимальный технологический эффект.

Это построение специальных систем различного уровня, создание систем обработки и хранения данных, централизация и виртуализация IT-сервиса, информационная защита и резервное копирование, решения по видеонаблюдению, IP-телефонии и многое другое [5].

## 2 Характеристика проекта для производственной практики

Требуется разработать Web-приложение, включающее в себя следующее:

- Страница авторизации. На странице необходимо задавать адрес входной ноды и опционально логин и пароль.
- Домашняя страница с основными параметрами кластера: число нод в кластере, число рабочих нод, число нод с проблемами, число нерабочих нод, информация по физическим дискам аналогична. Также необходимо включить общее число обрабатываемых запросов на кластере в секунду с разбивкой далее по put, get, exist, delete, общее место на дисках кластера, занятое данными место на дисках всего кластера.
- Страница с нодами. Страница должна отображать все ноды с именем и адресом, по каждой ноде её состояние – работает, есть проблемы, не работает; операции в секунду на ноде; занятое дисковое пространство. Перечень проблем на ноде: не доступен один из дисков, есть alien'ы, есть поврежденные БЛОБы, осталось свободного места менее 10%, виртуальная память, выделенная ноде, превышает доступную физическую память, нагрузка на CPU выше 90%. С данной страницы должна быть реализована возможность перехода на страницу с подробной информации о ноде.
- Страница с подробной информацией о ноде. Страница должна демонстрировать основные метрики ноды, параметры системы, на которой запущена нода – загрузка CPU, RAM, дескрипторы; список дисков на ноде с их состоянием, список виртуальных дисков внутри каждого диска. Также дать возможность просмотра целиком метрик ноды в виде таблицы.
- Страница с раскладкой виртуальных дисков. Страница должна отображать список виртуальных дисков с их доступностью в кластере.

### **3 Характеристика индивидуального задания для производственной практики**

В ходе практики предстоит выполнить следующие индивидуальные задачи:

1. Студент должен овладеть навыками командной работы, что позволит ему эффективно сотрудничать с другими разработчиками в рамках проекта.
2. Спроектировать и разработать клиент для взаимодействия с API [4] BOB. Данную задачу необходимо реализовать в первую очередь, поскольку клиент будет обеспечивать сервер данными о распределенной системе хранения данных.
3. Спроектировать собственный API [4], который позволит серверной части Web-приложения [2] передавать данные внешнему интерфейсу. Это API должно быть хорошо продумано, чтобы обеспечить удобный и эффективный доступ к данным BOB'a.
4. Разработать серверную часть Web-приложения [2], включив в него ранее разработанный клиент и API. Это будет являться основной частью работы, так как сервер должен обеспечивать функциональность авторизации, отображение информации о кластере, нодах, детальной информации о нодах и раскладке виртуальных дисков.
5. Провести тестирование итогового Web-приложения [2], чтобы убедиться в его работоспособности, надежности и соответствии поставленным задачам.

## **4 Описание выполнения задания**

### **4.1 Командная разработка**

Производственная практика включает в себя сотрудничество её участников для эффективной реализации поставленных задач в рамках проекта BOB.

В качестве системы управления версиями в рамках разрабатываемого проекта использовался Git. Для хранения проекта, а также для контроля статуса выполнения задач и четкого понимания сроков выполнения, был использован web-инструмент GitLab [6]. Участники проекта были распределены на 3 команды для равномерного распределения задач: команда отвечающая за интерфейс приложения, команда отвечающая за серверную часть приложения, а также команда дизайнеров.

В рамках вышеобозначенного инструмента GitLab была использована технология CI/CD [6] для снижения рисков и повышения надежности программного обеспечения.

## **4.2 Анализ используемых технологий**

В данном разделе проведен анализ технологий, использованных при разработке веб-интерфейса для мониторинга и управления кластером Боба

### **4.2.1 Rust**

В качестве языка программирования основного приложения используется мультипарадигменный язык Rust [7].

Rust — это язык программирования, ориентированный на безопасность, скорость и параллелизм. Его архитектура позволяет создавать программы, которые имеют высокую производительность и контроль на низком уровне, но с мощными высокоуровневыми абстракциями. Rust выполняет большую часть проверок и решений по управлению памятью во время компиляции, поэтому производительность во время выполнения программы не пострадает [8].

В контексте web-серверов, Rust становится особенно привлекательным выбором. Этот язык позволяет разработчикам создавать высокопроизводительные web-серверы с низкими задержками и малым потреблением памяти благодаря своей системе управления памятью. Благодаря сильным асинхронным возможностям и эффективной многозадачности, Rust идеально подходит для обработки большого числа одновременных запросов в web-приложениях.

### 4.2.2 OpenAPI

Спецификация OpenAPI (OAS) определяет стандартный, не зависящий от языка интерфейс для HTTP API, который позволяет как людям, так и компьютерам обнаруживать и понимать возможности сервиса без доступа к исходному коду, документации или с помощью проверки сетевого трафика. При правильном определении потребитель может понимать удаленный сервис и взаимодействовать с ним с минимальным количеством логики реализации.

Спецификация OpenAPI затем может быть использована инструментами создания документации для отображения API, инструментами генерации кода для создания серверов и клиентов на различных языках программирования, инструментами тестирования и многими другими вариантами использования [9].

### 4.2.3 Axum

Axum - это фреймворк для веб-приложений, который фокусируется на эргономике и модулярности [10].

Axum предлагает гибкую систему маршрутизации, промежуточное программное обеспечение, асинхронные обработчики, поддержку JSON и форм-кодированных тел запросов, а также интеграцию с другими Rust библиотеками.

### 4.2.4 TypeScript

В качестве языка программирования для написания интерфейса был выбран TypeScript.

Typescript — это язык программирования высокого уровня с открытым исходным кодом, разработанный Microsoft, который добавляет статическую типизацию с необязательными аннотациями типов в JavaScript. [11] Поскольку TypeScript является надмножеством JavaScript, все программы на JavaScript синтаксически корректны на TypeScript, но они могут не выполнять проверку типов по соображениям безопасности.



### 4.2.5 Next.js

В рамках языка TypeScript был выбран web-framework Next.js [12].

Next.js это фреймворк React для создания многофункциональных веб-приложений. Компоненты React используются для создания пользовательских интерфейсов, и Next.js для получения дополнительных функций и оптимизации.

Next.js также абстрагирует и автоматически настраивает инструменты, необходимые для React, такие как пакетирование, компиляция и многое другое.

### 4.2.6 React

React — это JavaScript-библиотека для разработки пользовательского интерфейса.

Компонентный подход, используемый в React, упрощает создание переиспользуемых элементов пользовательского интерфейса и облегчает управление состоянием приложения. Благодаря внутренней архитектуре, React обеспечивает высокую производительность и быстрое отображение компонентов на веб-странице.

## 4.3 Разработка серверной части приложения

Разработка серверной части согласовывалась с остальной частью команды.

## 4.4 OpenAPI Спецификация

Прежде чем приступить к фактическому написанию серверной части, было согласовано публичное API. В листинге 1 представлена часть согласованной API:

Листинг 1: Часть запросов согласно спецификации OpenAPI

```
1 openapi: 3.1.0
2 servers:
3   - url: "0.0.0.0:4000"
4     description: stub
5 info:
6   version: 0.1.0
```

```

7  title: Bob OpenAPI
8  description: Bob server OpenAPI schema
9  license:
10     name: Unlicensed
11 paths:
12     /api/login:
13         post:
14             operationId: login
15             summary: Authorize user
16             description: Connect user to the Node and authorize them if necessary
17             requestBody:
18                 content:
19                     application/json:
20                         schema:
21                             $ref: "#/components/schemas/BobConnectionData"
22                         examples:
23                             Example 1:
24                                 value:
25                                     hostname: "0.0.0.0:8000"
26                                     credentials:
27                                         login: archeoss
28                                         password: "1234"
29                             Example 2:
30                                 value:
31                                     hostname: "0.0.0.0:8000"
32             description: Data needed for the connection
33         responses:
34             "200":
35                 description: Ok
36                 headers:
37                     set-cookie:
38                         schema:
39                             type: string
40                         example: "sid=cT98Ao6mkYnkp4QJ48pxaKkB5ScYqulS9BxauJFQpeQ=4
hkEhWeG7nGSO/omtatgAxKS6uDgKcXnm2z/JW31MW9Ta39zFozivhLuMQGWD+
E9Pudlw5Hs22gMyWirmthaSw==; HttpOnly; SameSite=Strict; Secure; Path=/;
Expires=Fri, 14 Jul 2023 00:48:44 GMT"
41             description: Successfully authenticated. The session ID is
returned in a cookie named `sid`. You need to include this cookie in
subsequent requests.

```

```

42         "401":
43             description: Unauthorized
44         "404":
45             description: Server Not Found
46         parameters: []
47     parameters: []
48 /api/logout:
49     get:
50         summary: ""
51         operationId: logout
52         responses:
53             "200":
54                 description: OK
55                 description: Logout user from the Node
56                 parameters: []
57 /api/nodes:
58     get:
59         operationId: getNodes
60         summary: List of Nodes
61         description: Returns a list of all known nodes on cluster
62         parameters: []
63         responses:
64             "200":
65                 description: Node list
66                 content:
67                     application/json:
68                         schema:
69                             type: array
70                             items:
71                                 $ref: "#/components/schemas/Node"
72             "401":
73                 description: Unauthorized
74         security:
75             - ApiKeyAuth: []
76 /api/vdisks:
77     get:
78         operationId: getVdisks
79         summary: List of VDisks
80         description: Returns a list of all vdisks on cluster
81         parameters: []

```

```

82     responses:
83         "200":
84             description: Virtual disks list
85             content:
86                 application/json:
87                     schema:
88                         type: array
89                         items:
90                             $ref: "#/components/schemas/VDisk"
91         "401":
92             description: Unauthorized
93     security:
94         - ApiKeyAuth: []
95 /api/disks/count:
96     get:
97         operationId: getDisksCount
98         summary: List of Disks
99         description: Returns a list with count of physical disks per status
100    parameters: []
101    responses:
102        "200":
103            description: Disk count per status
104            content:
105                application/json:
106                    schema:
107                        $ref: "#/components/schemas/DiskCount"
108                    examples:
109                        Example 1:
110                            value:
111                                good: 5
112                                bad: 10
113                                offline: 50
114        "401":
115            description: Unauthorized
116    security:
117        - ApiKeyAuth: []
118    parameters: []
119    ...

```

#### 4.4.1 Клиентская часть

В клиентском участке серверной части был определен трейт ‘API’ - листинг 2 для определения методов, доступных на кластере ВОВ.

Листинг 2: Трейт API

```
1 #[async_trait]
2 pub trait Api<C: Send + Sync> {
3     fn poll_ready(&self, _cx: &mut Context
4         ) -> Poll<Result<(), Box<dyn Error + Send + Sync + 'static>>> {
5         Poll::Ready(Ok(()))
6     }
7
8     /// Return directory of alien
9     async fn get_alien_dir(&self, context: &C) -> Result<GetAlienDirResponse,
10         ApiError>;
11
12     /// Returns the list of disks with their states
13     async fn get_disks(&self, context: &C) -> Result<GetDisksResponse,
14         ApiError>;
15
16     /// Get metrics
17     async fn get_metrics(&self, context: &C) -> Result<GetMetricsResponse,
18         ApiError>;
19
20     /// Returns a list of known nodes
21     async fn get_nodes(&self, context: &C) -> Result<GetNodesResponse,
22         ApiError>;
23
24     /// Returns a partition info by ID
25     async fn get_partition(
26         &self,
27         v_disk_id: i32,
28         partition_id: String,
29         context: &C,
30     ) -> Result<GetPartitionResponse, ApiError>;
31
32     /// Returns a list of partitions
33     async fn get_partitions(&self, v_disk_id: i32, context: &C,
34     ) -> Result<GetPartitionsResponse, ApiError>;
```

```

31
32     /// Returns count of records of this on node
33     async fn get_records(
34         &self,
35         v_disk_id: i32,
36         context: &C,
37     ) -> Result<GetRecordsResponse, ApiError>;
38
39     /// Returns directories of local replicas of vdisk
40     async fn get_replicas_local_dirs(
41         &self,
42         v_disk_id: i32,
43         context: &C,
44     ) -> Result<GetReplicasLocalDirsResponse, ApiError>;
45
46     /// Get space info
47     async fn get_space_info(&self, context: &C) -> Result<
48         GetSpaceInfoResponse, ApiError>;
49
50     /// Returns information about self
51     async fn get_status(&self, context: &C) -> Result<GetStatusResponse,
52         ApiError>;
53
54     /// Returns a vdisk info by ID
55     async fn get_v_disk(&self, v_disk_id: i32, context: &C) -> Result<
56         GetVDiskResponse, ApiError>;
57
58     /// Returns a list of vdisks
59     async fn get_v_disks(&self, context: &C) -> Result<GetVDisksResponse,
60         ApiError>;
61
62     /// Returns server version
63     async fn get_version(&self, context: &C) -> Result<GetVersionResponse,
64         ApiError>;
65
66     /// Returns configuration of the node
67     async fn get_configuration(&self, context: &C) -> Result<
68         GetConfigurationResponse, ApiError>;
69 }

```

## 4.4.2 Методы авторизации

Для авторизации был определен соответствующий метод, описанный в листинге 3:

Листинг 3: Метод Авторизации

```
1 pub async fn login(  
2     State(store): State<Arc<RwLock<BobStore<Hostname>>>>,&br/>3     mut auth: AuthContext<Hostname, BobClient, SocketBobMemoryStore>,&br/>4     Extension(request_timeout): Extension<RequestTimeout>,&br/>5     Json(bob): Json<BobConnectionData>,&br/>6 ) -> AxumResult<StatusCode> {  
7     tracing::info!("post /login : {:?}", &bob);  
8     let hostname = bob.hostname.clone();  
9  
10    let Ok(bob_client) = BobClient::try_new(bob, request_timeout).await else  
11    {  
12        tracing::warn!("Couldn't create client");  
13        return Err(StatusCode::UNAUTHORIZED.into());  
14    };  
15    let Ok(res) = bob_client.probe().await else {  
16        return Err(StatusCode::UNAUTHORIZED.into());  
17    };  
18    tracing::info!("received {res} from BOB");  
19    if res == StatusCode::OK {  
20        if let Err(err) = auth.login(&bob_client).await {  
21            tracing::warn!("Couldn't login the user. Err: {err}, User: {  
22                bob_client:?}");  
23            return Err(StatusCode::UNAUTHORIZED.into());  
24        };  
25        store.write().await.insert(hostname, bob_client);  
26        tracing::info!("AUTHORIZATION SUCCESSFUL");  
27        tracing::info!("Logged in as {:?}", &auth.current_user);  
28    }  
29    Ok(res)  
30 }
```

### 4.4.3 Методы публичного API

Для взаимодействия с графическим интерфейсом были написаны методы, определенные ранее в API. Один из таких методов представлен в листинге 4:

Листинг 4: Метод публичного API

```
1 pub async fn get_nodes_count(Extension(client): Extension<BobClient>) -> Json
  <NodeCount> {
2   tracing::info!("get /nodes/count : {:?}", client);
3
4   let mut metrics: FuturesUnordered<_> = client
5     .cluster()
6     .map(move |node| {
7       let handle = node.clone();
8       tokio::spawn(async move { handle.get_metrics().await })
9     })
10    .collect();
11
12   let mut count = NodeCount::new();
13
14   let mut counter = 0;
15   while let Some(res) = metrics.next().await {
16     if let Ok(Ok(client::GetMetricsResponse::Metrics(metrics))) = res {
17       tracing::info!("#{counter}: metrics received successfully");
18       let metrics = Into::<TypedMetrics>::into(metrics);
19       if is_bad_node(&metrics) {
20         count[Bad] += 1;
21       } else {
22         count[Good] += 1;
23       }
24     } else {
25       tracing::warn!("#{counter}: couldn't receive metrics from node");
26       count[Offline] += 1;
27     }
28     counter += 1;
29   }
30   tracing::info!("total nodes per status count: {count:?}");
31
32   Json(count)
33 }
```



## 4.5 Демонстрация работы веб-приложения

На рисунке 1 изображена страница авторизации, где пользователь должен обязательно ввести адрес и порт, а также логин и пароль:

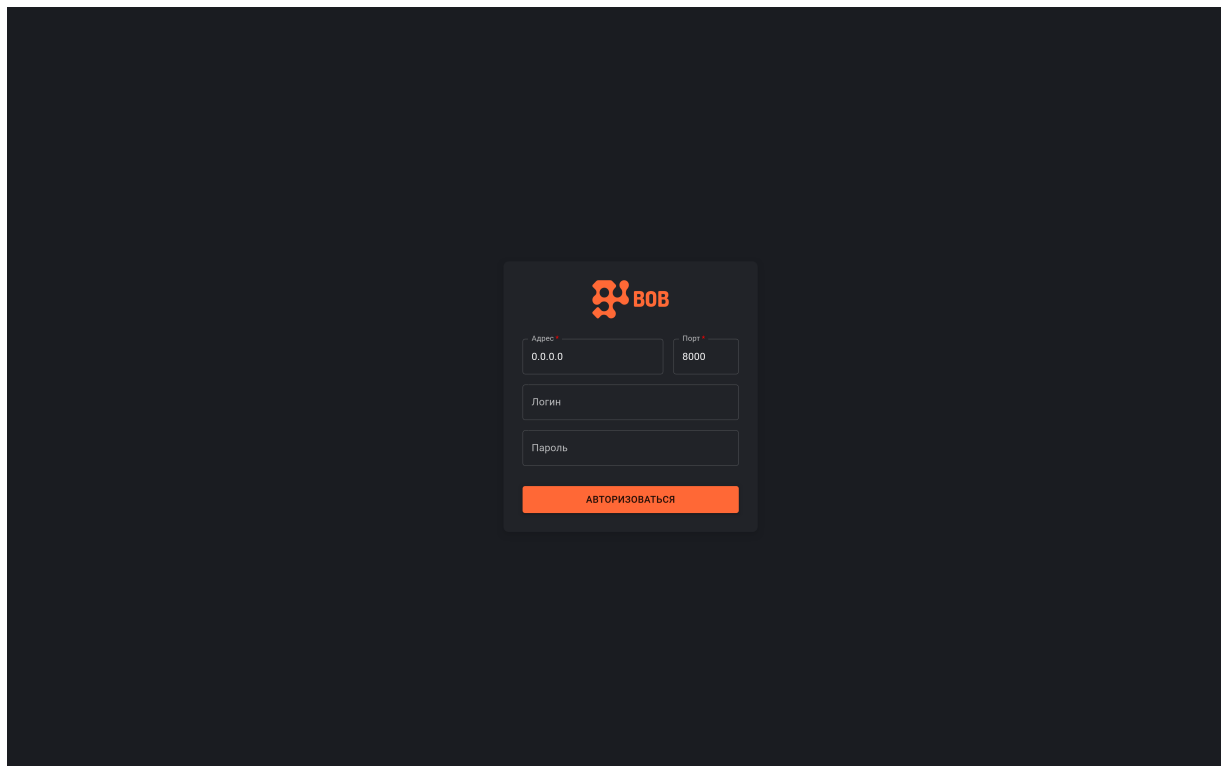
The image shows a dark-themed web application interface for authentication. At the top center is the VOB logo, consisting of an orange icon and the text 'VOB'. Below the logo are four input fields: 'Адрес' (Address) with the value '0.0.0.0', 'Порт' (Port) with the value '8000', 'Логин' (Login), and 'Пароль' (Password). At the bottom of the form is an orange button labeled 'АВТОРИЗОВАТЬСЯ' (Authenticate).

Рисунок 1 – Страница авторизации

На рисунке 2 изображена страница мониторинга ресурсов кластера BOB, которая обновляется в реальном времени, обновление можно как отключить, так и установить частоту обновления информации:

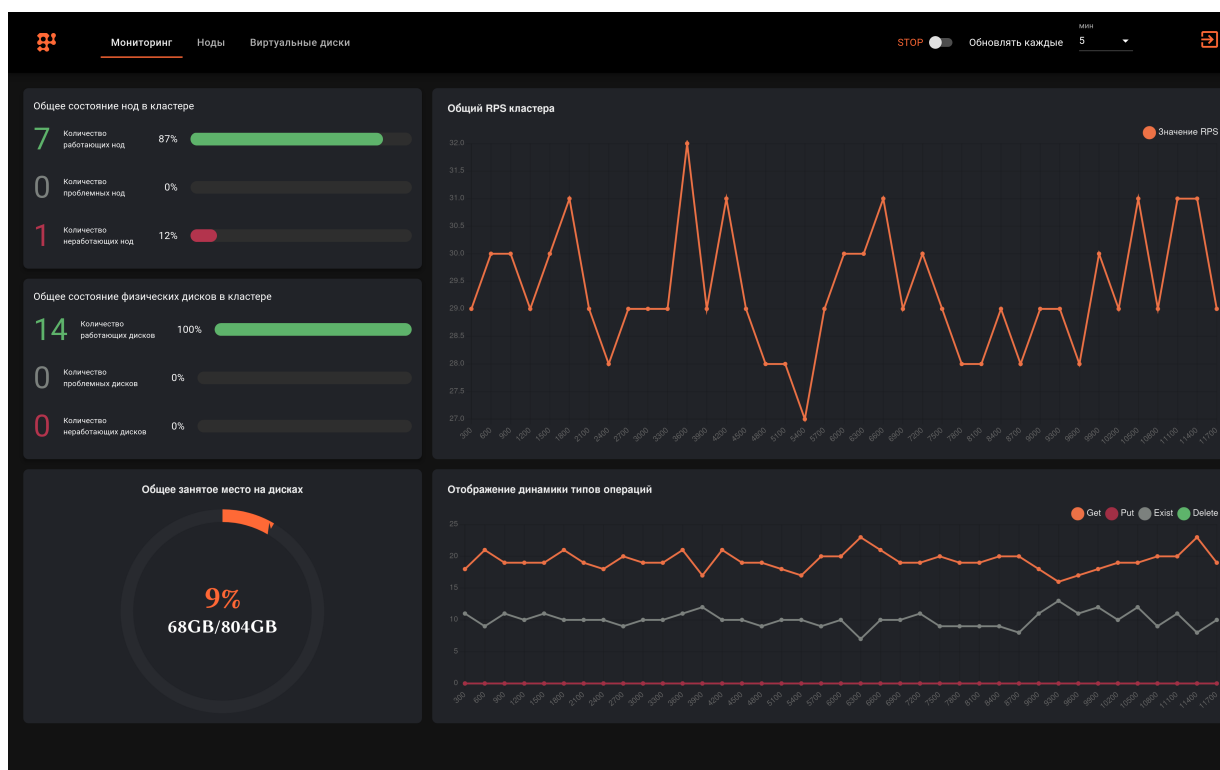


Рисунок 2 – Домашняя страница с основными параметрами кластера

На рисунке 3 изображена информация по всем нодам:

Имя ноды	Адрес	Состояние	Занятое место	RPS	Алиены	Поврежденные BLOB
node_1	192.168.17.11.20000	● Работающая	5GB / 115GB	0	0	0
node_7	192.168.17.17.20000	● Работающая	5GB / 115GB	0	0	0
node_6	192.168.17.16.20000	● Работающая	5GB / 115GB	0	0	0
node_0	192.168.17.10.20000	● Работающая	5GB / 115GB	0	0	0
node_2	192.168.17.12.20000	● Работающая	5GB / 115GB	0	0	0
node_5	192.168.17.15.20000	● Не работающая	0 / 0	0	0	0
node_3	192.168.17.13.20000	● Работающая	5GB / 115GB	0	0	0
node_4	192.168.17.14.20000	● Работающая	5GB / 115GB	0	0	0

Рисунок 3 – Страница с нодами

На рисунке 4 изображена детальная информация по выбранной ноде из списка предоставленного на рисунке 3:

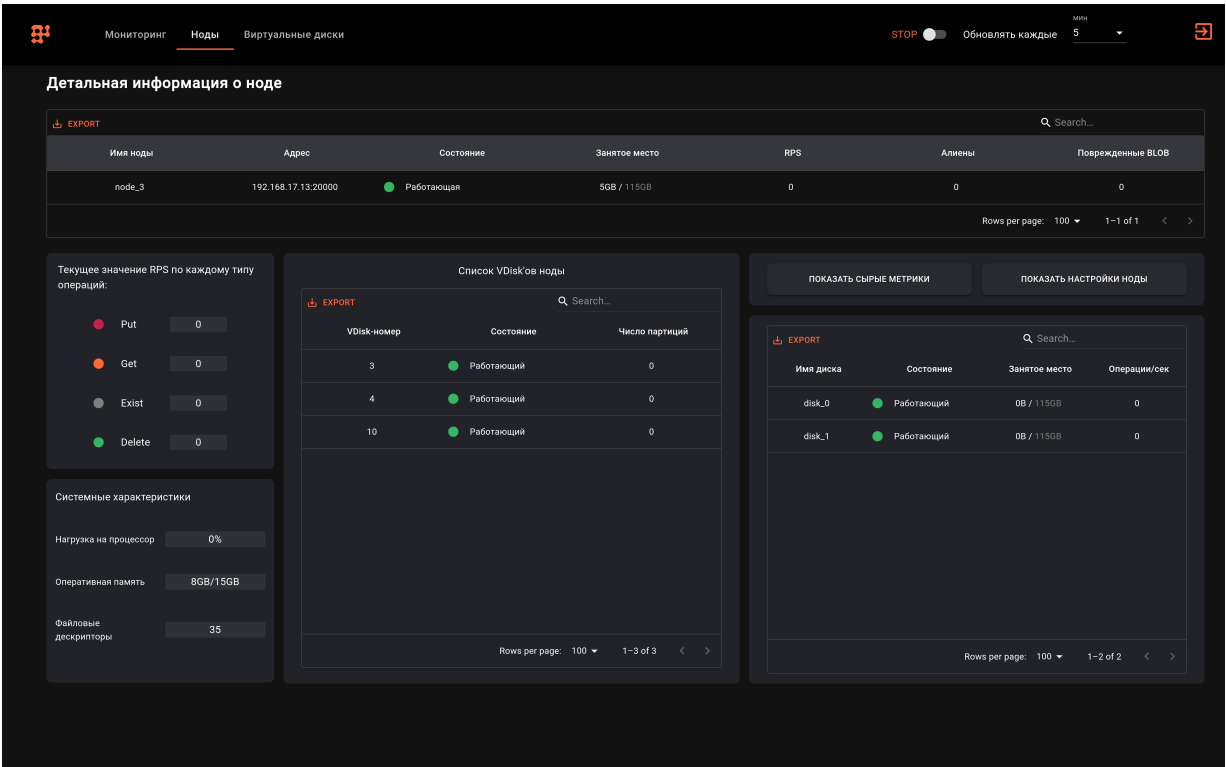


Рисунок 4 – Страница с подробной информацией о ноде

На рисунке 5 изображена информация по всем виртуальным дискам:

EXPORT Search...

Диск-номер	Копии на нодах	Доступность	Состояние
8	node_2, node_1	2/2	Работающий
6	node_7	1/1	Работающий
10	node_3, node_4	2/2	Работающий
2	node_0, node_2	2/2	Работающий
7	node_5, node_6	1/2	Проблемный
5	node_4	1/1	Работающий
11	node_2	1/1	Работающий
0	node_0, node_1	2/2	Работающий
1	node_0	1/1	Работающий
3	node_3	1/1	Работающий
4	node_4, node_2, node_3	3/3	Работающий
9	node_5	0/1	Не работающий

Rows per page: 100 1 - 12 of 12

Рисунок 5 – Страница с раскладкой виртуальных дисков

## ЗАКЛЮЧЕНИЕ

В ходе производственной практики были приобретены навыки в разработке веб-приложений и open source-проектов, а также было успешно разработано веб-приложение, предназначенное для эффективной работы с настройкой и управлением кластером BOB.

Также были выполнены следующие индивидуальные задачи:

- 1) получить навыки командной работы;
- 2) спроектировать и разработать клиент для взаимодействия с API [4] Боба;
- 3) спроектировать собственный API [4];
- 4) разработать серверную часть веб-приложения, интегрируя в него ранее разработанный клиент и API [4];
- 5) провести тестирование итогового Web-приложения.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. BOB - Distributed BLOB storage [Электронный ресурс]. Режим доступа: <https://github.com/qoollo/bob> (дата обращения: 01.07.2023).
2. What is a Web Application? [Электронный ресурс]. Режим доступа: <https://www.stackpath.com/edge-academy/what-is-a-web-application> (дата обращения: 01.07.2023).
3. What is open source software? [Электронный ресурс]. Режим доступа: <https://opensource.com/resources/what-open-source> (дата обращения: 01.07.2023).
4. What is an API? [Электронный ресурс]. Режим доступа: <https://www.ibm.com/topics/api> (дата обращения: 01.07.2023).
5. ООО «СитиСофт» [Электронный ресурс]. Режим доступа: [https://www.nic.ru/help/chto-takoe-subd\\_8580.html](https://www.nic.ru/help/chto-takoe-subd_8580.html) (дата обращения: 01.07.2023).
6. Cowell C., Lotz N., Timberlake C. Automating DevOps with GitLab CI/CD Pipelines: Build efficient CI/CD pipelines to verify, secure, and deploy your code using real-life examples. Packt Publishing, 2023. URL: <https://books.google.ru/books?id=X36rEAAAQBAJ>.
7. Rust Programming Language [Электронный ресурс]. Режим доступа: <https://www.rust-lang.org/> (дата обращения: 01.07.2023).
8. Klabnik S., Nichols C. The Rust Programming Language, 2nd Edition. No Starch Press, 2023. URL: <https://books.google.ru/books?id=a8l9EAAAQBAJ>.
9. OpenAPI Specification [Электронный ресурс]. Режим доступа: <https://swagger.io/specification/> (дата обращения: 01.07.2023).

10. axum - Rust | Docs.rs [Электронный ресурс]. Режим доступа: <https://docs.rs/axum/latest/axum/> (дата обращения: 01.07.2023).
11. Microsoft TypeScript: the JavaScript we need, or a solution looking for a problem? [Электронный ресурс]. Режим доступа: <https://arstechnica.com/information-technology/2012/10/microsoft-typescript-the-javascript-we-need-or-a-solution-looking> (дата обращения: 01.07.2023).
12. Next.js Documentation [Электронный ресурс]. Режим доступа: <https://nextjs.org/docs> (дата обращения: 01.07.2023).