МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное

учреждение высшего образования

«КАЗАНСКИЙ (ПРИВОЛЖСКИЙ) ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт информационных технологий и интеллектуальных систем

**ОТЧЕТ**

**о прохождении производственной (проектной) практики**

Обучающийся \_ Али Хезам Абдулрахман Хезам\_11-011 \_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_

(ФИО, группа) (подпись)

Руководитель практики от КФУ:

\_\_\_\_\_\_ Кошарский И.Е., ассистент кафедры программной инженерии \_\_\_\_\_\_\_\_\_

(ФИО, должность)

Оценка за практику: \_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись руководителя практики)

Дата сдачи отчета: 12.07.2023

Казань – 2023 г.

**Содержание**

[***Введение***](#_heading=h.bv7dvbbxbmhe)

[***CRUD-приложение с мониторингом***](#_heading=h.6w4empabbgk3)

***Prometheus***

***Mysql-exporter***

***Grafana***

[***Заключение***](#_heading=h.paisyxj3wrf8)

[***Список использованных источников***](#_heading=h.6irp9ybc91a2)

# Введение

Производственная практика является одним из компонентов и составной частью учебного процесса студентов бакалавриата. Данный вид практики выполняет функции общепрофессиональной подготовки в части формирования необходимых компетенций и подготовки студентов к будущей профессиональной деятельности.

Дата начала практики: 29.06.2023 г.

Дата окончания практики: 12.07.2023 г.

**Место прохождения практики:** Лаборатория 42

**Цель прохождения практики:** учимся работать с мониторингом в веб-разработке

Для достижения поставленной цели практики были сформулированы следующие задачи:

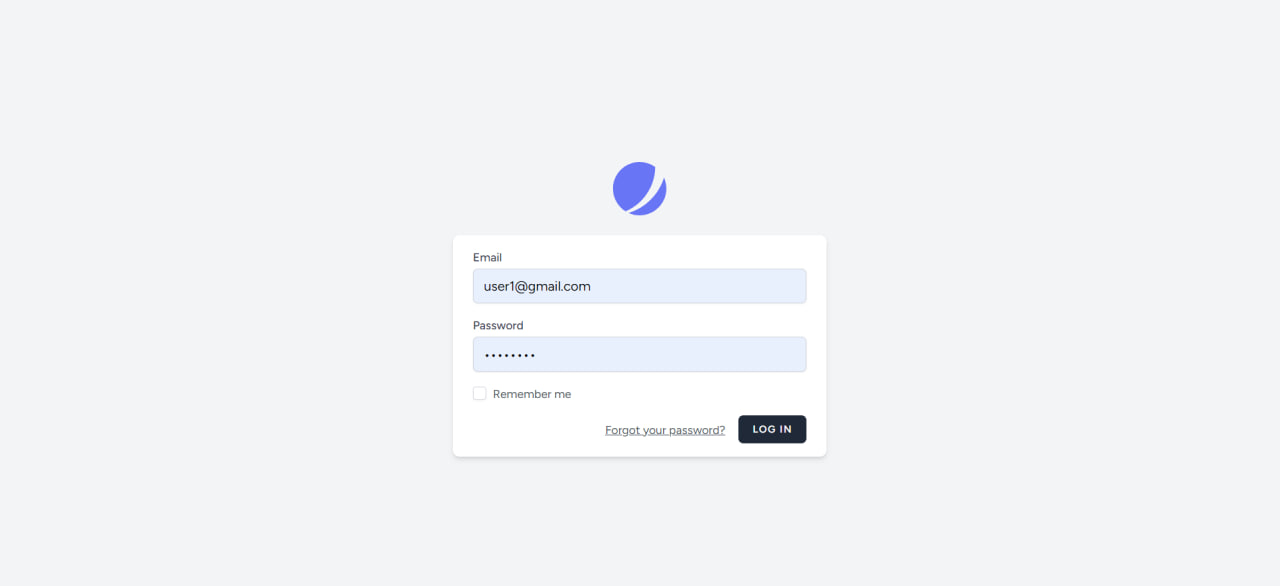
Проект мониторинга на prometheus/victoriametrics стеков и grafana.

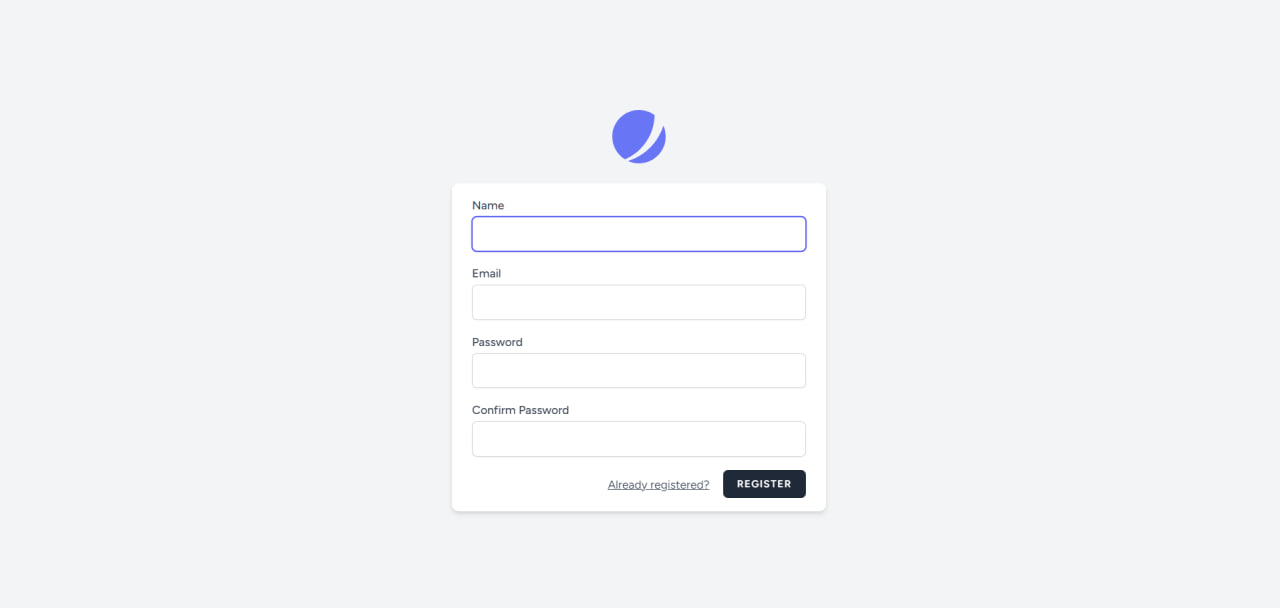
Должно быть

1. Описание компонентов подсистемы .
2. Полный перечень того, что монитором и чем.
3. Автоматизация на развертывание exporter в инфраструктуре + описание встраивания exporter в контейнеры приложений, если такие присутствуют.
4. Стенд в виде мониторинга в контейнерах, вашего приложения

CRUD-приложение с мониторингом

CRUD - это аббревиатура, пришедшая из мира компьютерного программирования и обозначающая четыре функции, которые считаются необходимыми для реализации приложения постоянного хранения: создание, чтение, обновление и удалени .

* вход в систему и регистрация



*Отобразить список ресурса*

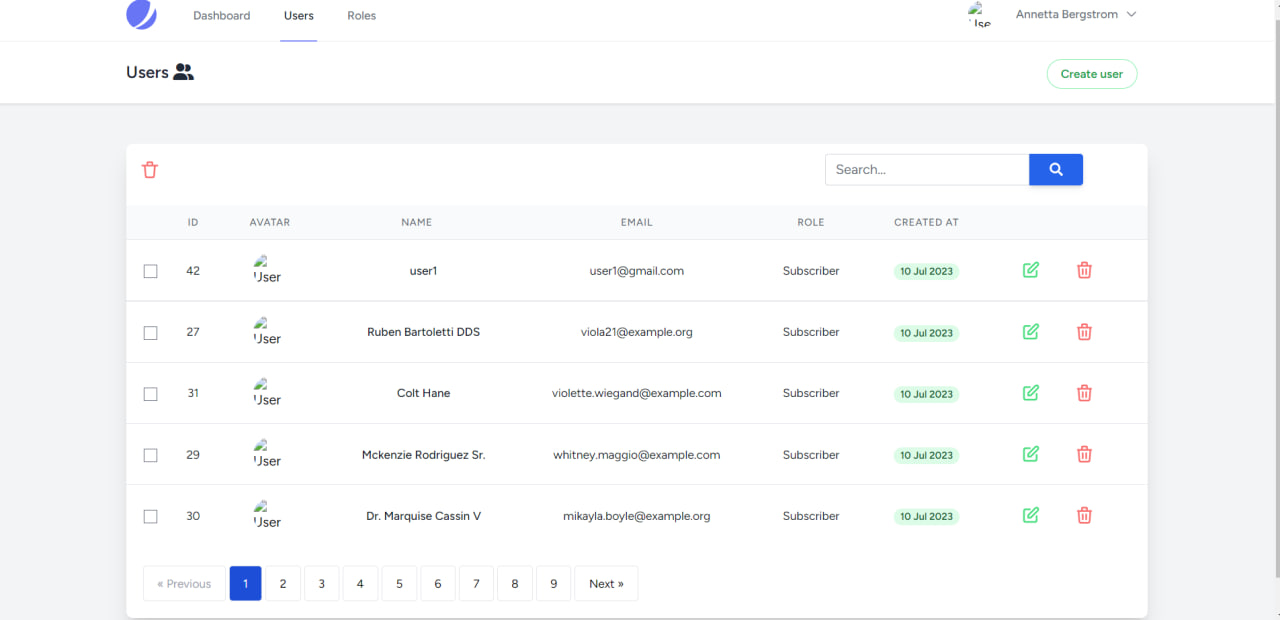
**public function** index(Request $request)  
{  
 $term = $request->input(**'term'**);  
  
 $users = User::*with*(**'role'**)  
 ->when($term, **function** ($query, $term) {  
 $query->where(**'name'**, **'LIKE'**, **'%'** . $term . **'%'**)  
 ->orWhere(**'email'**, **'LIKE'**, **'%'** . $term . **'%'**);  
 })  
 ->latest()  
 ->paginate(5);  
  
 **return** Inertia::**render**(**'Users/Index'**, *compact*(**'users'**));  
}

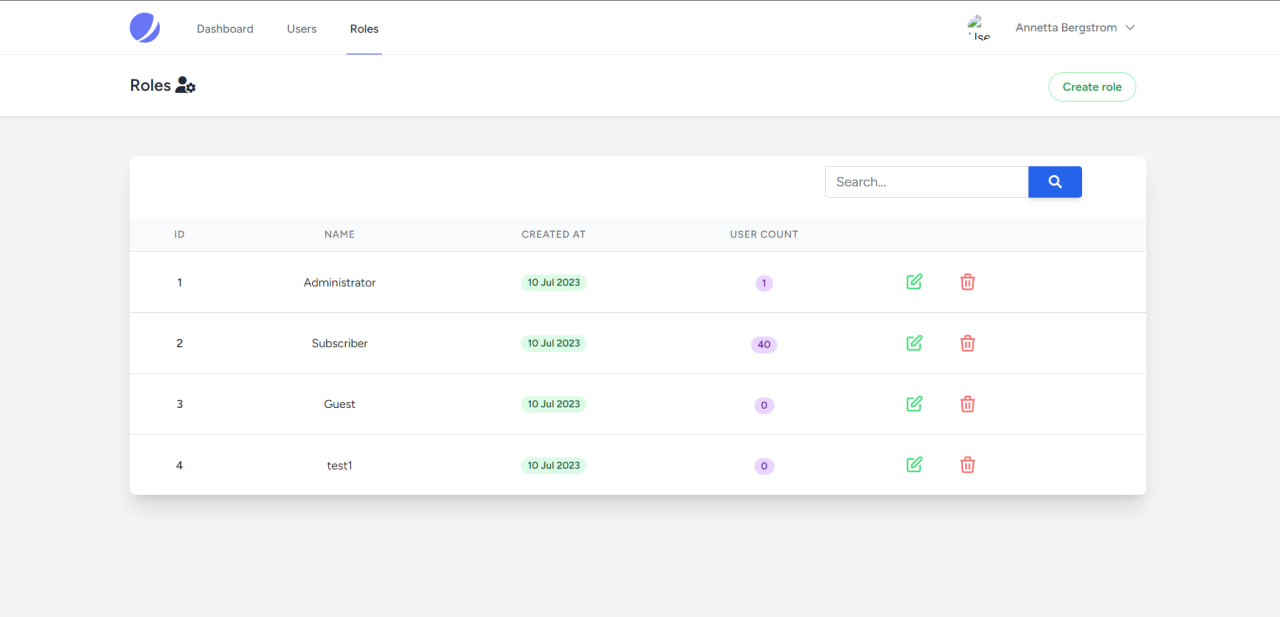
*Отобразите форму для редактирования указанного ресурса.*

**public function** edit(**string** $id)  
{  
 $user = User::*findOrFail*($id);  
  
 **return** Inertia::**render**(**'Users/Edit'**, *compact*(**'user'**));  
}  
  
*Обновите указанный ресурс в хранилище*

**public function** update(Request $request, **string** $id)  
{  
 User::*where*(**'id'**, $id)->update($request->all());  
  
 **return** redirect(**'/users'**)->with(**'success'**, **'User has been updated!'**);  
}

*Сохраните вновь созданный ресурс в хранилище.***public function** store(UserStoreRequest $request)  
{  
 $validatedData = $request->validated();  
  
 **if** ($request->hasFile(**'avatar'**)) {  
 $avatarPath = $request->file(**'avatar'**)->store(**'public/avatars'**);  
 $validatedData[**'avatar'**] = Storage::**url**($avatarPath);  
 }  
  
 $validatedData[**'password'**] = Hash::**make**($validatedData[**'password'**]);  
 User::*create*($validatedData);  
  
 **return** redirect()->route(**'users.index'**)->with(**'success'**, **'User has been created!'**);  
}





Метрики

Сделал метрики маршрута для конечной точки Prometheus

Route::middleware([RequestsMiddleware::class])->group(function () {

Route::get('/metrics',[MyMetricsController::class, 'myMetrics']);

});

Route::middleware(['auth:sanctum', config('jetstream.auth\_session'), 'verified'])->group(function () {

Route::resource('/users', UserController::class);

Route::resource('/roles', RoleController::class);

});

В классе My MetricsController имеет функцию my Metrics

public function myMetrics(Request $request)

{

DB::connection()->enableQueryLog();

$collectorRegistry = app(CollectorRegistry::class);

Показатель использования процессора

$cpuUsage = sys\_getloadavg()[0]; // Retrieves the average system load for the last minute

$gauge = $collectorRegistry->getOrRegisterGauge(

'spa',

'cpu\_usage\_percentage',

'CPU usage percentage'

);

$gauge->set($cpuUsage);

показатель использования памяти

$memoryUsage = memory\_get\_usage(true);

$gauge = $collectorRegistry->getOrRegisterGauge(

'spa',

'memory\_usage\_bytes',

'Memory usage in bytes'

);

$gauge->set($memoryUsage);

Подсчитайте количество зарегистрированных пользователей

$usersRegistered = User::count();

$gauge = $collectorRegistry->getOrRegisterGauge(

'spa',

'users\_registered\_total',

'Total number of registered users'

);

$gauge->set($usersRegistered);

Подсчитайте количество зарегистрированных ролей

$usersRegistered = Role::count();

$gauge = $collectorRegistry->getOrRegisterGauge(

'spa',

'roles\_registered\_total',

'Total number of registered roles'

);

$gauge->set($usersRegistered);

Извлеките показатели из реестра

$renderer = new RenderTextFormat();

$result = $renderer->render($collectorRegistry->getMetricFamilySamples());

return response($result, 200)->header('Content-Type', RenderTextFormat::MIME\_TYPE);

}

В классе RequestsMiddlewarejava функция обрабатывает

public function handle(Request $request, Closure $next)

{

$startTime = microtime(true);

$response = $next($request);

$duration = microtime(true) - $startTime;

$path = $request->getPathInfo();

$method = $request->getMethod();

$statusCode = $response->getStatusCode();

$content = $response->getContent();

$requestCounter = $this->registry->getOrRegisterCounter(

'spa',

'request\_count',

'Total number of requests',

['path', 'method', 'status\_code']

);

$requestCounter->incBy(1, [$path, $method, (string) $statusCode]);

$requestSize = $this->registry->getOrRegisterHistogram(

'spa',

'request\_size\_bytes',

'Request size in bytes',

['path', 'method']

);

$requestSize->observe(strlen($content), [$path, $method]);

return $response;

}

сделал это в файле docker

FROM php:8.2.0-fpm

# Install system dependencies

RUN apt-get update && apt-get install -y \

libzip-dev \

zip \

libpq-dev \

libfreetype6-dev \

libjpeg62-turbo-dev \

libpng-dev \

curl \

&& docker-php-ext-install zip pdo\_mysql pdo\_pgsql

# Install GD extension

RUN docker-php-ext-configure gd --with-freetype --with-jpeg \

&& docker-php-ext-install -j$(nproc) gd

# Install Node.js and npm

RUN curl -fsSL https://deb.nodesource.com/setup\_14.x | bash -

RUN apt-get install -y nodejs

# Install Composer

RUN curl -sS https://getcomposer.org/installer | php -- --install-dir=/usr/local/bin --filename=composer

WORKDIR /var/www/html

# Copy application files

COPY . .

# Set max\_execution\_time

RUN echo "php\_value[max\_execution\_time] = 120" >> /usr/local/etc/php/conf.d/docker-php-max-execution-time.ini

# Install PHP dependencies

RUN composer install

# Install Node.js dependencies and build assets

RUN npm install

в файле Docker compose для пользования всеми услугами

services:

spa:

build:

context: .

dockerfile: Dockerfile

container\_name: spa

volumes:

- .:/var/www/html

depends\_on:

- mysql

- prometheus

- mailpit

mysql:

image: mysql:8.0

container\_name: mysql

ports:

- '3306:3306'

environment:

MYSQL\_ROOT\_PASSWORD: '${DB\_PASSWORD}'

MYSQL\_ROOT\_HOST: "%"

MYSQL\_DATABASE: '${DB\_DATABASE}'

MYSQL\_PASSWORD: '${DB\_PASSWORD}'

volumes:

- spa\_mysql\_data:/var/lib/mysql

mysql-exporter:

image: prom/mysqld-exporter

container\_name: mysql-exporter

ports:

- "9104:9104"

environment:

DATA\_SOURCE\_NAME: "root:pass123@(mysql:3306)/spa"

command:

- "--config.my-cnf=/etc/my.cnf"

volumes:

- ./monitoring/my.cnf:/etc/my.cnf

prometheus:

image: prom/prometheus

container\_name: prometheus

ports:

- "9090:9090"

volumes:

- ./monitoring:/etc/monitoring

- ./monitoring/scrape\_job.yml:/etc/monitoring/scrape\_job.yml

command:

- --config.file=/etc/monitoring/scrape\_job.yml

depends\_on:

- mysql-exporter

grafana:

image: grafana/grafana

container\_name: grafana

ports:

- "3000:3000"

depends\_on:

- prometheus

nginx:

image: nginx:latest

container\_name: nginx

ports:

- '8000:8000'

volumes:

- .:/var/www/html

- ./monitoring/nginx.conf:/etc/nginx/conf.d/default.conf

depends\_on:

- spa

mailpit:

image: axllent/mailpit

container\_name: mailpit

ports:

- '1025:1025'

- '8025:8025'

volumes:

spa\_mysql\_data:

* Конфигурация экспортера MySQL

A close-up of a text

Description automatically generated

* A computer screen shot of a computer code

  Description automatically generatedКонфигурация сервера
* Экспортер MySQL и конфигурация Prometheus

A computer screen shot of a computer

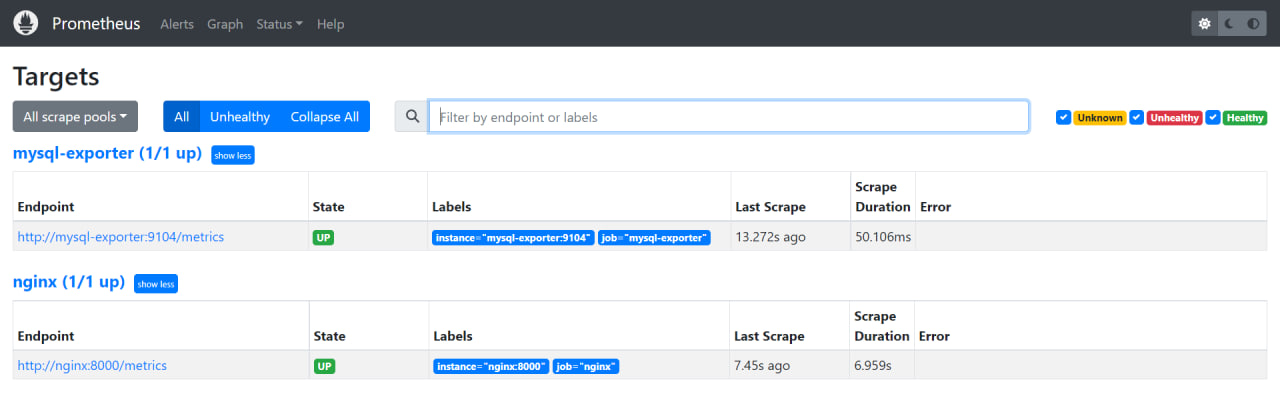
Description automatically generated

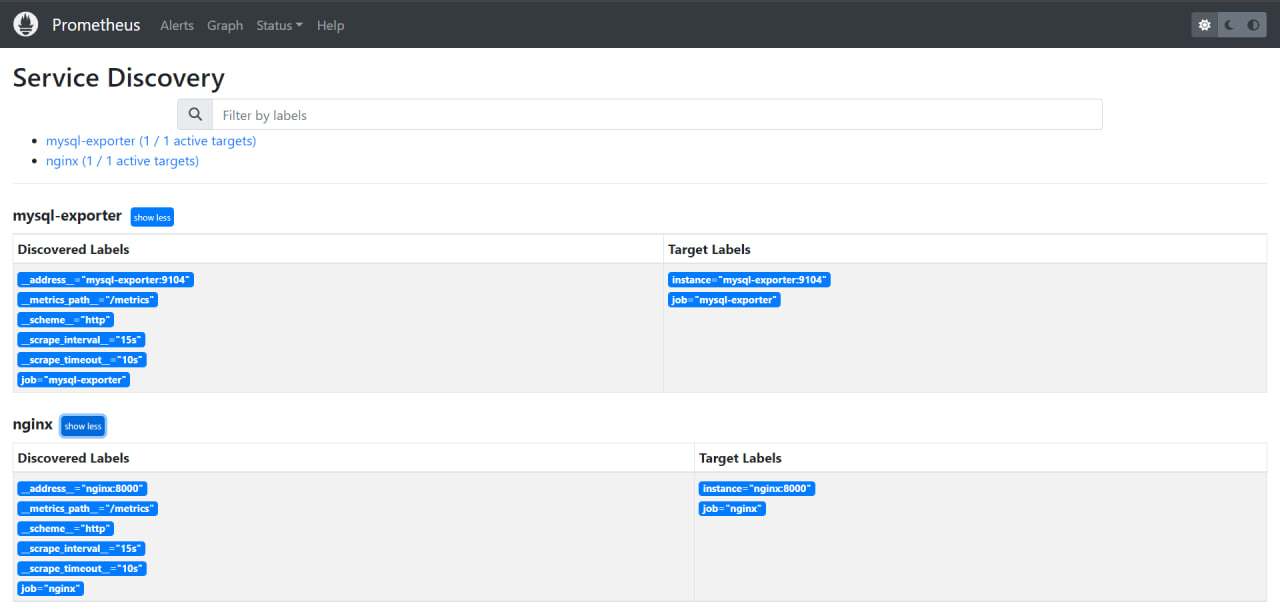
* Отображать Метрики в моем приложении

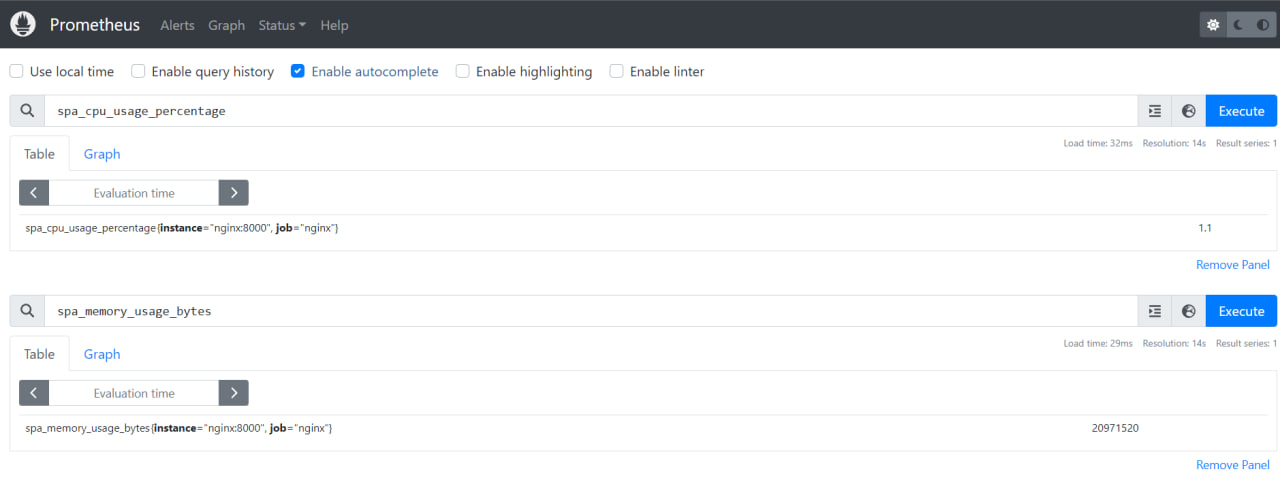
A screenshot of a computer program

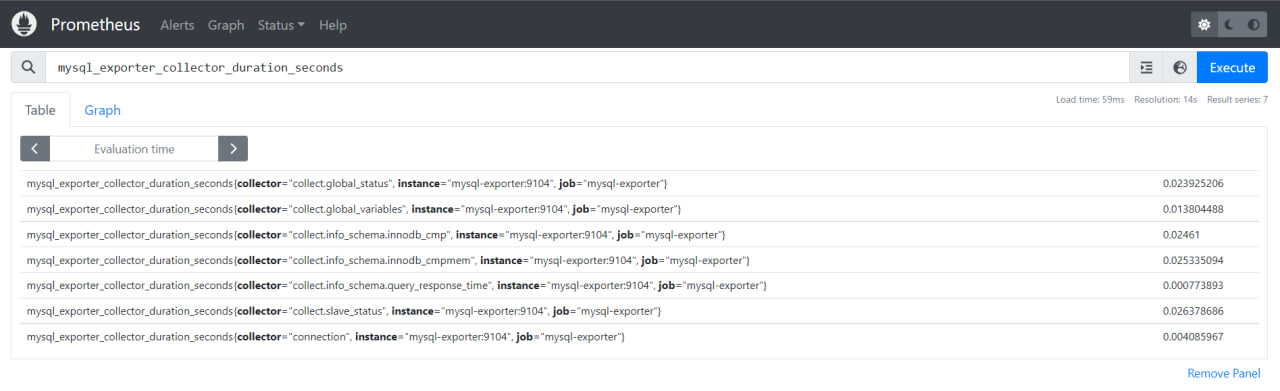
Description automatically generated

**Prometheus**

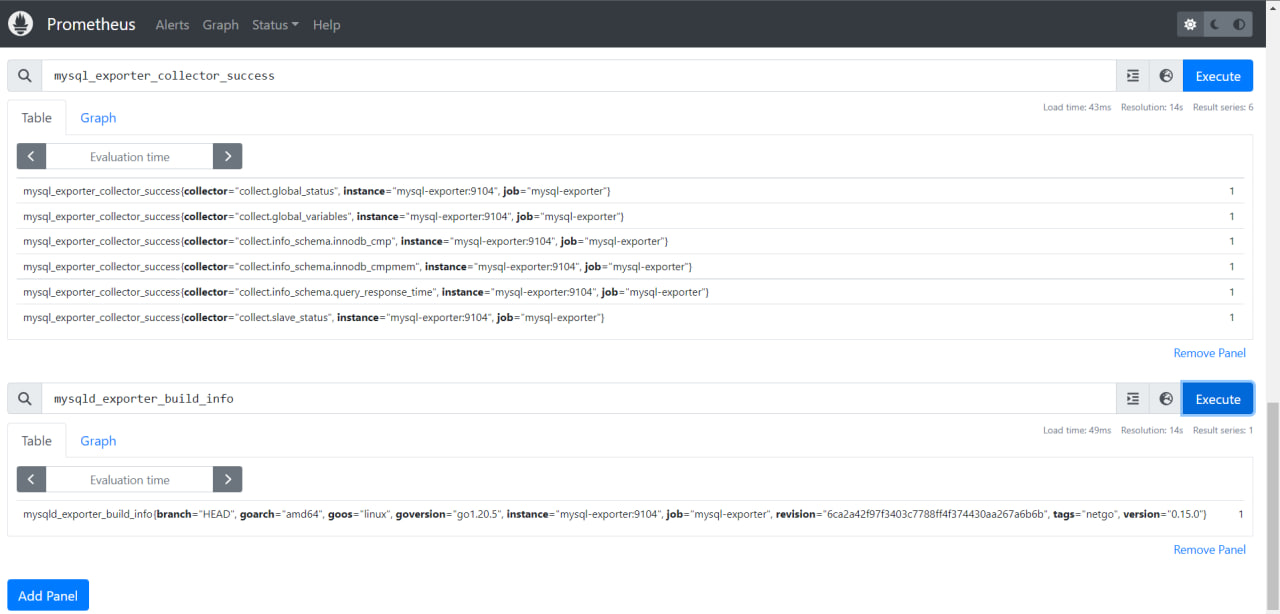
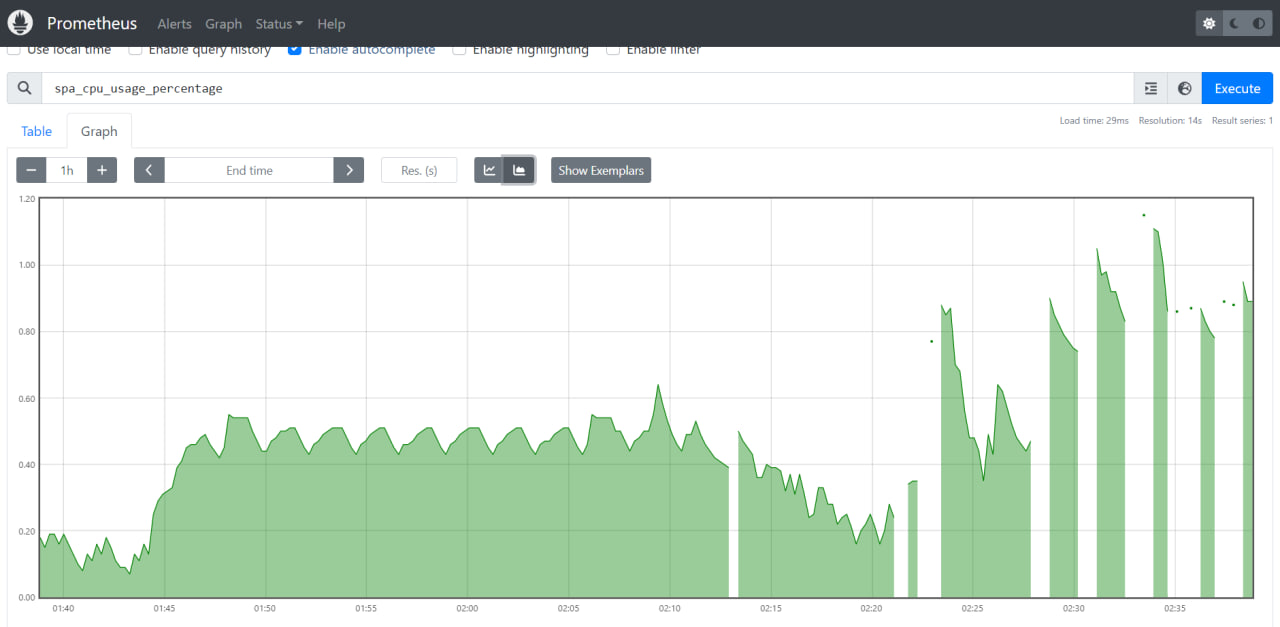
****







A screenshot of a graph

Description automatically generated

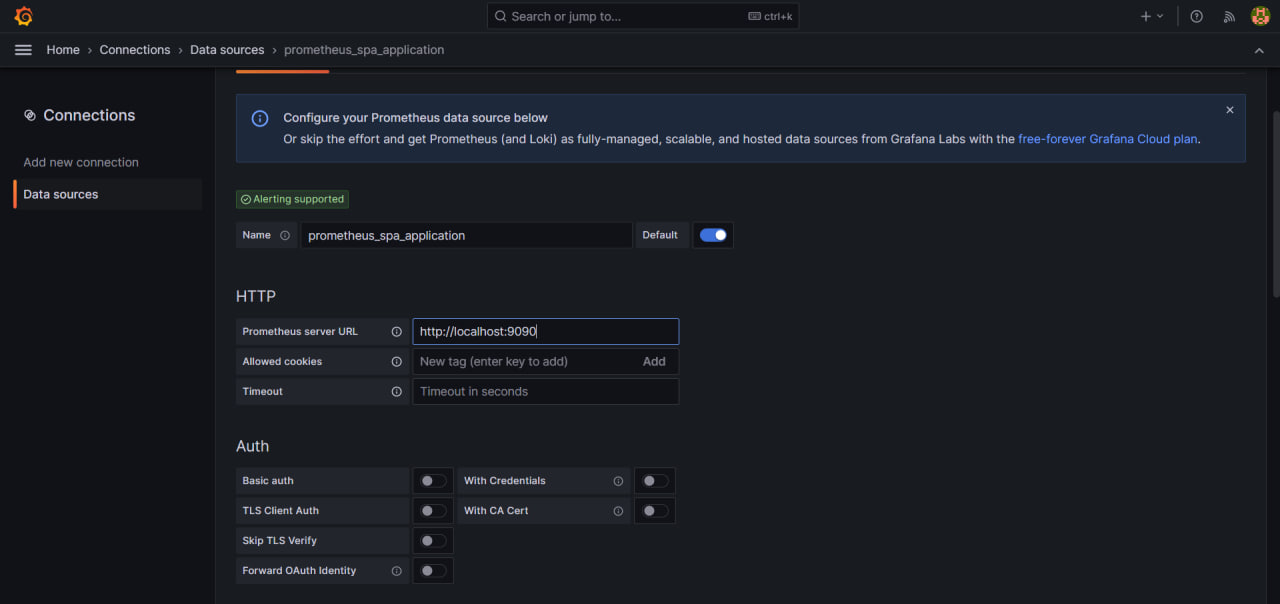
# A screenshot of a graph Description automatically generated

# 

# Mysql-exporter

# 

# Grafana



A screenshot of a computer

Description automatically generated

# 

# 

# 

# Заключение

Данная производственная практика была направлена на формирование следующих компетенций:

|  |  |
| --- | --- |
| Шифр компетенции | Расшифровка  приобретаемой компетенции |
| УК-3 | Способен осуществлять социальное взаимодействие и реализовывать свою роль в команде |
| ПК-7 | Владение навыками использования операционных систем, сетевых технологий, средств разработки программного интерфейса, применения языков и методов формальных спецификаций, систем управления базами данных |
| ПК-8 | Владение навыками использования различных технологий разработки программного обеспечения |

Получил много полезной информации в области разработки веб-приложений, освоил ее и работал над решением задачи, и это было немного сложно

# Список использованных источников

1. <https://docs.docker.com/>
2. <https://grafana.com/docs/>
3. <https://prometheus.io/docs/introduction/overview/>
4. <https://grafana.com/oss/prometheus/exporters/mysql-exporter/>
5. <https://docs.docker.com/compose/>
6. <https://github.com/>
7. <https://stackoverflow.com/>