

## Инвариантные СР

### ИСП 1

#### Проектирование CRUD для серверного веб-приложения

Реализован REST API для выполнения операций CRUD в базе данных MongoDB.

Приложение реализовано с использованием Express, bodyParser, Mongoose.

Реализованы следующие операции:

- ❖ GET /products - получение списка продуктов
- ❖ POST /products/new - добавление нового продукта
- ❖ POST /products/update/:id - обновление информации о продукте по ID
- ❖ POST /products/delete/:id - удаление информации о продукте по ID

При отправке запросов для добавления и обновления продукта информация о нём передаётся в теле POST-запроса.

Сервер возвращает ответ - статус выполнения операции в формате JSON. Пример ответа сервера (добавление данных):

```
{  
  "status": "success"  
}
```

Пример ответа сервера на GET-запрос:

```
[  
  {  
    "amount": {  
      "number": 32,  
      "unit": "kg"  
    },  
    "_id": "60e4268f46a5bd73485414f0",  
    "name": "Apples",  
    "type": "Fruits"  
  },  
  {  
    "amount": {  
      "number": 16.25,  
      "unit": "kg"  
    },  
    "_id": "60e42ad546a5bd73485414f1",
```

```
"name": "Bananas",
"type": "Fruits"
},
{
  "amount": {
    "number": 24.5,
    "unit": "kg"
  },
  "_id": "60e42b1046a5bd73485414f2",
  "name": "Strawberries",
  "type": "Berries"
}
]
```

Файл index.js: <https://pastebin.com/pJHcwSRT>

Файл package.json: <https://pastebin.com/HzjyVLx>

---

## ИСП 2

### Проектирование приложения на основе фреймворка Symfony

Приложение реализует API для выполнения арифметических операций.

Маршруты описаны в файле routes.yaml.

Контроллер реализован в файле CalcController.php.

Ответ сервера при выполнении запроса по маршруту /calc/add/2/3:

```
{
  "result": 5
}
```

Ответ сервера при выполнении запроса по маршруту /calc/substract/5/7:

```
{
  "result": -2
}
```

Ответ сервера при выполнении запроса по маршруту /calc/multiply/7/5:

```
{
  "result": 35
}
```

Ответ сервера при выполнении запроса по маршруту /calc/divide/5/7:

```
{
  "result": 0.7142857142857143
}
```

Файл routes.yaml: <https://pastebin.com/bTNVZcgZ>

Файл CalcController.php: <https://pastebin.com/48ZeMECc>

---

## ИСП 3

### Разработка БД на MongoDB

Для разработки базы данных на MongoDB можно использовать MongoDB Cloud.

Этапы разработки базы данных:

1. создание базы данных
2. создание коллекций
3. добавление данных
4. MongoDB не требует определения схемы документа, что позволяет вносить изменения в структуру документа при добавлении или редактировании данных.

Этапы создания базы данных в MongoDB Cloud:

1. создание организации (organization)
2. создание проекта (project)
3. создание кластера (cluster) и базы данных с помощью MongoDB Atlas
4. добавление данных с помощью MongoDB Atlas, MongoDB Compass или веб-приложения

The screenshot displays the MongoDB Cloud web interface for a collection named 'products'. At the top, it shows 'COLLECTION SIZE: 98B', 'TOTAL DOCUMENTS: 1', and 'INDEXES TOTAL SIZE: 20KB'. Below this is a navigation bar with tabs: 'Find', 'Indexes', 'Schema Anti-Patterns', 'Aggregation', and 'Search Indexes'. The 'Find' tab is active, showing a filter input with the text 'FILTER {"filter": "example"}'. To the right of the filter are buttons for 'Find' and 'Reset'. Below the filter, it says 'QUERY RESULTS 1-3 OF 3'. The results are displayed in a list of three documents, each with a red ID and a blue name. The first document is an Apple, the second is a Banana, and the third is a Strawberry. Each document has a 'type' field set to 'Fruits' and an 'amount' field with a 'number' and 'unit' sub-field.

```
mongodb-test.products
COLLECTION SIZE: 98B  TOTAL DOCUMENTS: 1  INDEXES TOTAL SIZE: 20KB
Find  Indexes  Schema Anti-Patterns  Aggregation  Search Indexes
FILTER {"filter": "example"}  Find  Reset
QUERY RESULTS 1-3 OF 3
{
  "_id": ObjectId("60e4268f46a5bd73485414f0"),
  "name": "Apples",
  "type": "Fruits",
  "amount": {
    "number": 32,
    "unit": "kg"
  }
}
{
  "_id": ObjectId("60e42ad546a5bd73485414f1"),
  "name": "Bananas",
  "type": "Fruits",
  "amount": {
    "number": 16.25,
    "unit": "kg"
  }
}
{
  "_id": ObjectId("60e42b1046a5bd73485414f2"),
  "name": "Strawberries",
  "type": "Berries",
  "amount": {
    "number": 24.5,
    "unit": "kg"
  }
}
```

---

## ИСП 4

Подготовка виртуального сервера или деплой-платформы для публикации веб-ресурса на основе Ghost.js

Ghost.js можно установить с помощью Docker-контейнера.

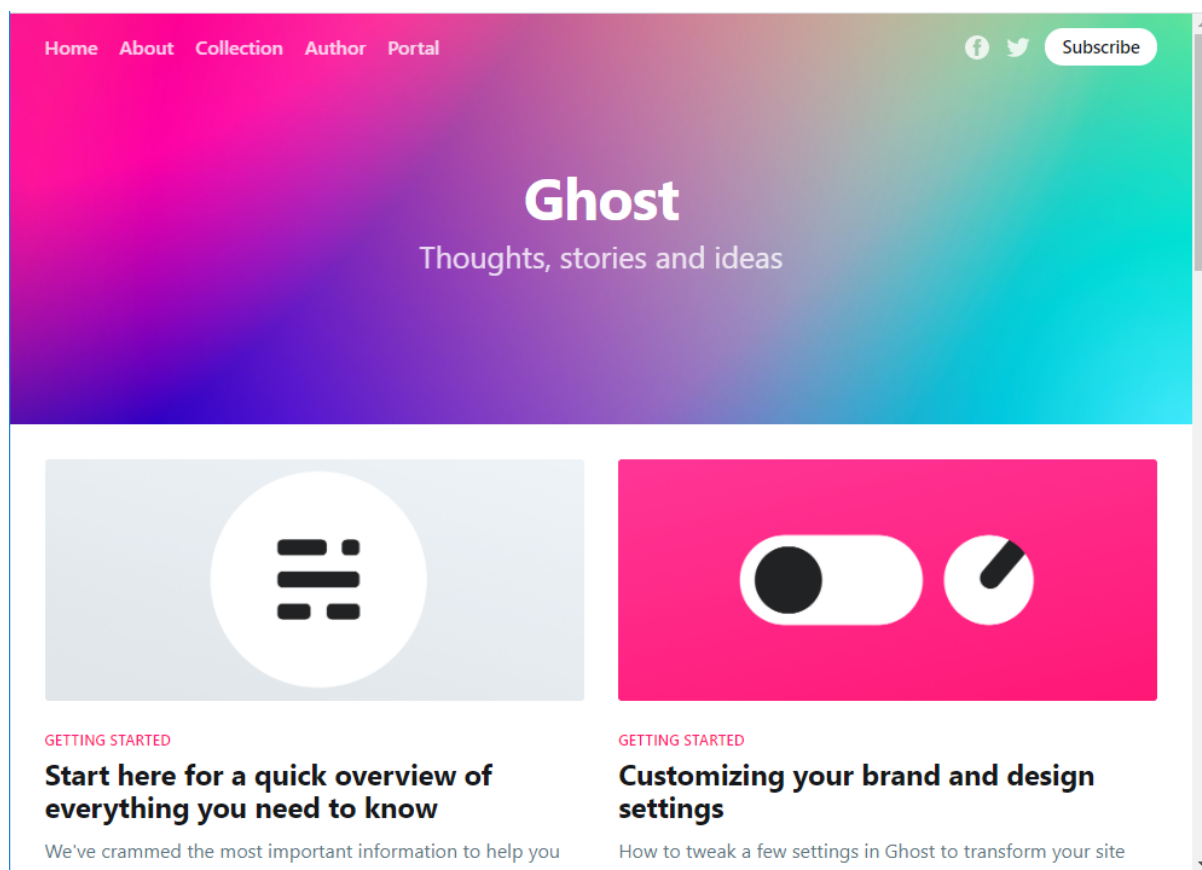
Для установки контейнера необходимо выполнить команду:

```
docker pull ghost
```

Для запуска Ghost необходимо выполнить команду:

```
docker run -d --name some-ghost -e url=http://localhost:3001 -p 3001:2368 ghost
```

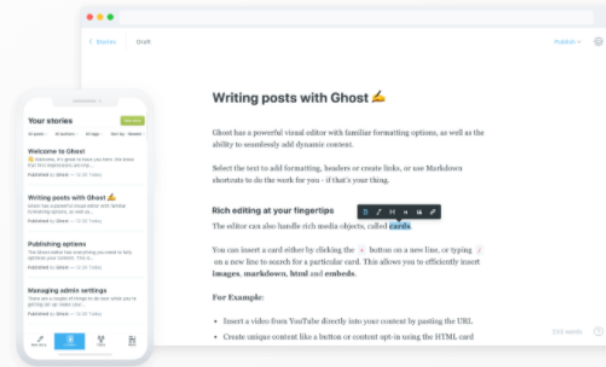
После этого Ghost будет доступен по адресу `server_ip:3001`.



Панель администратора доступна по адресу `server_ip:3001/ghost`.

# Welcome to Ghost!

All over the world, people have started **2,000,000+** incredible sites with Ghost. Today, we're starting yours.



Create your account →