

Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО
ITMO University

Отчёт по лабораторной работе 4

По дисциплине Web-программирование

Тема работы Отчёт по лабораторной работе 4

Обучающийся Мартынюк Алексей Петрович

Факультет факультет инфокоммуникационных технологий

Группа К3320

Направление подготовки 11.03.02 Инфокоммуникационные технологии и системы связи

Образовательная программа Программирование в инфокоммуникационных системах

Обучающийся	_____	_____	<u>Мартынюк А. П.</u>
	(дата)	(подпись)	(Ф.И.О.)

Руководитель	_____	_____	<u>Марченко Е.В.</u>
	(дата)	(подпись)	(Ф.И.О.)

СОДЕРЖАНИЕ

ВВЕДЕНИЕ.....	3
1 WEB-приложение на php.....	4
1.1 Проектирование таблиц в MySQL.....	4
1.2 Data-models.....	5
1.3 UI.....	9
1.4 Тестирование взаимодействия с системой.....	11
1.5 docker-compose.yaml.....	13
3 WEB-сервер.....	15
3.1 Реализация.....	15
3.2 Запуск web-server.....	16
ЗАКЛЮЧЕНИЕ.....	17

ВВЕДЕНИЕ

Цель работы:

Получить базовые навыки программирования на php. Создать веб-приложение со следующим функционалом: создание заказа, авторизация пользователей. Данные сохранять в бд MySQL Server. Написать веб-сервер для обработки запросов, который берет порт из переменных окружения.

1 WEB-приложение на php

1.1 Проектирование таблиц в MySQL

С помощью миграций были созданы следующие таблицы в бд для поддержания функционала авторизации пользователя, просмотра товаров и создания корзины(заказа):

```
CREATE TABLE IF NOT EXISTS `users` (  
  `id` INT PRIMARY KEY AUTO_INCREMENT,  
  `email` VARCHAR(100) NOT NULL,  
  `password_hash` VARCHAR(255) NOT NULL,  
  `first_name` VARCHAR(50) NOT NULL,  
  `last_name` VARCHAR(50) NOT NULL,  
  `created_at` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,  
  `updated_at` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,  
  
  UNIQUE INDEX `idx_users_email` (`email`)  
);
```

Рисунок 1 — Таблица users

```
CREATE TABLE IF NOT EXISTS `user_addresses` (  
  `id` INT PRIMARY KEY AUTO_INCREMENT,  
  `user_id` INT NOT NULL,  
  `address` VARCHAR(255) NOT NULL,  
  `postcode` VARCHAR(6) NOT NULL,  
  `created_at` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,  
  `updated_at` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,  
  
  CONSTRAINT `fk_paddresses_user_id` FOREIGN KEY (`user_id`) REFERENCES `users` (`id`) ON DELETE CASCADE  
);
```

Рисунок 2 — Таблица user_addresses

```
CREATE TABLE IF NOT EXISTS `products` (  
  `id` INT PRIMARY KEY AUTO_INCREMENT,  
  `name` VARCHAR(100) NOT NULL,  
  `price` INT NOT NULL,  
  `stock` INT NOT NULL DEFAULT 0,  
  `image_url` VARCHAR(255),  
  `created_at` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,  
  `updated_at` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP  
);  
  
CREATE FULLTEXT INDEX `product_name_ft_index` ON `products` (`name`);
```

Рисунок 3 — Таблица products

```
CREATE TABLE IF NOT EXISTS `categories` (  
  `id` INT PRIMARY KEY AUTO_INCREMENT,  
  `name` VARCHAR(100) NOT NULL,  
  `description` TEXT,  
  `created_at` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,  
  `updated_at` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,  
  
  UNIQUE INDEX `idx_categories_name` (`name`)  
);
```

Рисунок 4 — Таблица categories

```
CREATE TABLE IF NOT EXISTS `product_category` (
  `product_id` INT NOT NULL,
  `category_id` INT NOT NULL,

  PRIMARY KEY (`product_id`, `category_id`),

  CONSTRAINT `fk_pcategories_product_id` FOREIGN KEY (`product_id`) REFERENCES `products` (`id`) ON DELETE CASCADE,
  CONSTRAINT `fk_pcategories_category_id` FOREIGN KEY (`category_id`) REFERENCES `categories` (`id`) ON DELETE CASCADE
);
```

Рисунок 5 — Таблица product_category

```
CREATE TABLE IF NOT EXISTS `orders` (
  `id` INT PRIMARY KEY AUTO_INCREMENT,
  `user_id` INT NOT NULL,
  `address_id` INT NOT NULL,
  `status` ENUM ('progress', 'completed', 'canceled') NOT NULL DEFAULT 'progress',
  `comment` TEXT,
  `created_at` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
  `updated_at` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,

  CONSTRAINT `fk_orders_user_id` FOREIGN KEY (`user_id`) REFERENCES `users` (`id`) ON DELETE CASCADE,
  CONSTRAINT `fk_orders_address_id` FOREIGN KEY (`address_id`) REFERENCES `user_addresses` (`id`) ON DELETE CASCADE
);
```

Рисунок 6 — Таблица orders

```
CREATE TABLE IF NOT EXISTS `order_items` (
  `id` INT PRIMARY KEY AUTO_INCREMENT,
  `order_id` INT NOT NULL,
  `product_id` INT NOT NULL,
  `amount` INT NOT NULL,
  `price_per_unit` INT NOT NULL,
  `created_at` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
  `updated_at` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,

  CONSTRAINT `fk_oitems_order_id` FOREIGN KEY (`order_id`) REFERENCES `orders` (`id`) ON DELETE CASCADE,
  CONSTRAINT `fk_oitems_product_id` FOREIGN KEY (`product_id`) REFERENCES `products` (`id`) ON DELETE RESTRICT
);
```

Рисунок 7 — Таблица order_items

1.2 Data-models

Для доступа к объектам базы данных были написаны соответствующие Repository в папке data с необходимыми методами: category.php, order.php, product.php, user.php. Ниже представлен пример файла user.php.

```

class UserRepository {
    private PDO $connection;

    Codeium: Refactor | Explain | Generate Function Comment
    public function __construct(PDO $connection) {
        $this->connection = $connection;
    }

    Codeium: Refactor | Explain | Generate Function Comment
    public function insert(User $user): bool {
        $stmt = $this->connection->prepare("
            INSERT INTO users (email, password_hash, first_name, last_name)
            VALUES (:email, :password_hash, :first_name, :last_name)
        ");

        return $stmt->execute([
            ':email' => $user->email,
            ':password_hash' => $user->password_hash,
            ':first_name' => $user->first_name,
            ':last_name' => $user->last_name
        ]);
    }

    Codeium: Refactor | Explain | Generate Function Comment
    public function getById(int $id): ?User {
        $stmt = $this->connection->prepare("
            SELECT *
            FROM users
            WHERE id = :id
        ");
        $stmt->bindValue(':id', $id, PDO::PARAM_INT);
        $stmt->execute();
        $row = $stmt->fetchObject();

        if ($row) {
            $user = new User();
            $user->id = $row->id;
            $user->email = $row->email;
            $user->password_hash = $row->password_hash;
            $user->first_name = $row->first_name;
            $user->last_name = $row->last_name;
            $user->created_at = new DateTime($row->created_at);
            $user->updated_at = new DateTime($row->updated_at);

            return $user;
        }
        return null;
    }
}

```

Рисунок 8 — UserRepository(методы insert и getById)

```

public function getByEmail(string $email): ?User {
    $stmt = $this->connection->prepare("
        SELECT *
        FROM users
        WHERE email = :email
    ");

    $stmt->execute([':email' => $email]);
    $row = $stmt->fetchObject();

    if ($row) {
        $user = new User();
        $user->id = $row->id;
        $user->email = $row->email;
        $user->password_hash = $row->password_hash;
        $user->first_name = $row->first_name;
        $user->last_name = $row->last_name;
        $user->created_at = new DateTime($row->created_at);
        $user->updated_at = new DateTime($row->updated_at);

        return $user;
    }

    return null;
}

```

Рисунок 9 — UserRepository(метод getByEmail)

```

public function getAdressesById(int $id): array {
    $stmt = $this->connection->prepare("
        SELECT *
        FROM user_addresses
        WHERE user_id = :user_id
    ");

    $stmt->bindValue(':user_id', $id, PDO::PARAM_INT);
    $stmt->execute();

    $rawResult = $stmt->fetchAll();

    $result = [];
    foreach ($rawResult as $row) {
        $address = new Address();
        $address->id = $row['id'];
        $address->user_id = $row['user_id'];
        $address->address = $row['address'];
        $address->postcode = $row['postcode'];
        $address->created_at = new DateTime($row['created_at']);
        $address->updated_at = new DateTime($row['updated_at']);

        $result[] = $address;
    }

    return $result;
}

```


Рисунок 10 — UserRepository(метод getAdressesById)

Подключение к базе данных и функция проверки авторизации пользователя в системе были вынесены в отдельный файл с конфигурацией config.php.

```
1  <?php
2
3  session_start();
4
5  $dsn = 'mysql:host=mysql;dbname=lab4';
6  $username = 'root';
7  $password = 'password';
8
9  try {
10     $pdo = new PDO($dsn, $username, $password);
11     $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
12 } catch (PDOException $e) {
13     echo 'Connection failed: ' . $e->getMessage();
14     exit;
15 }
16
17 Codeium: Refactor | Explain | Generate Function Comment
18 function is_auth(): bool {
19     return isset($_SESSION['user_id']);
20 }
```

Рисунок 11 — config.php

Также в отдельный файл password-utils.php была вынесена логика взаимодействия с паролем пользователя: преобразование строки в биты, инвертация битов, преобразование из битов в строку.

```
<?php

Codeium: Refactor | Explain | Generate Function Comment
function invertStringBits($string) {
    return invertBits(stringToBinary($string));
}

Codeium: Refactor | Explain | Generate Function Comment
function invertBits($binaryString) {
    $inverted = '';
    for ($i = 0; $i < strlen($binaryString); $i++) {
        $inverted .= ($binaryString[$i] === '0') ? '1' : '0';
    }
    return $inverted;
}

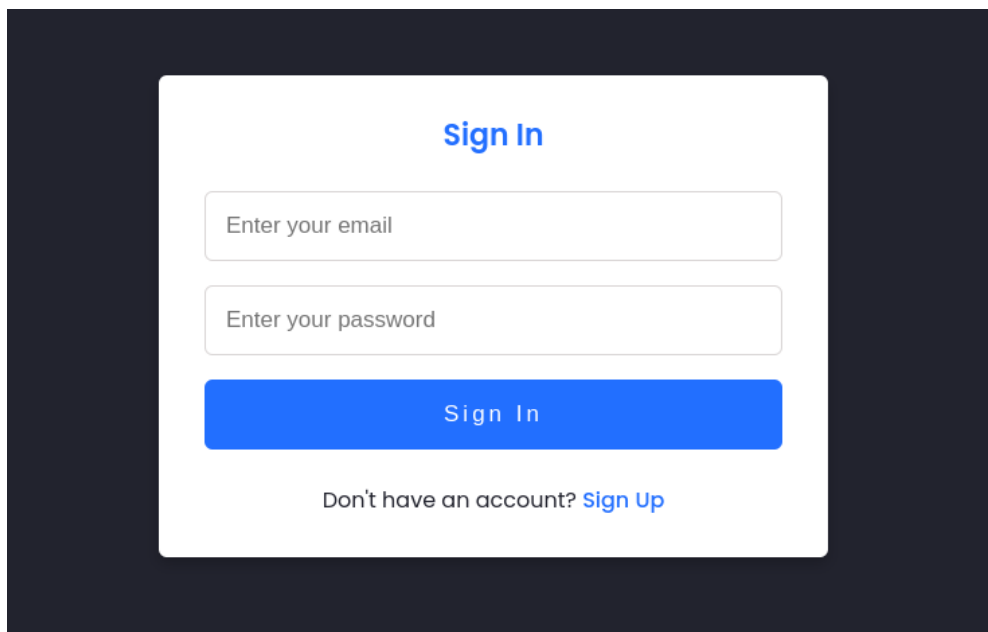
Codeium: Refactor | Explain | Generate Function Comment
function stringToBinary($string) {
    $binaryString = '';
    for ($i = 0; $i < strlen($string); $i++) {
        $binaryChar = decbin(ord($string[$i]));
        $binaryString .= str_pad($binaryChar, 8, '0', STR_PAD_LEFT);
    }
    return trim($binaryString);
}
```


Рисунок 12 — password-utils.php

1.3 UI

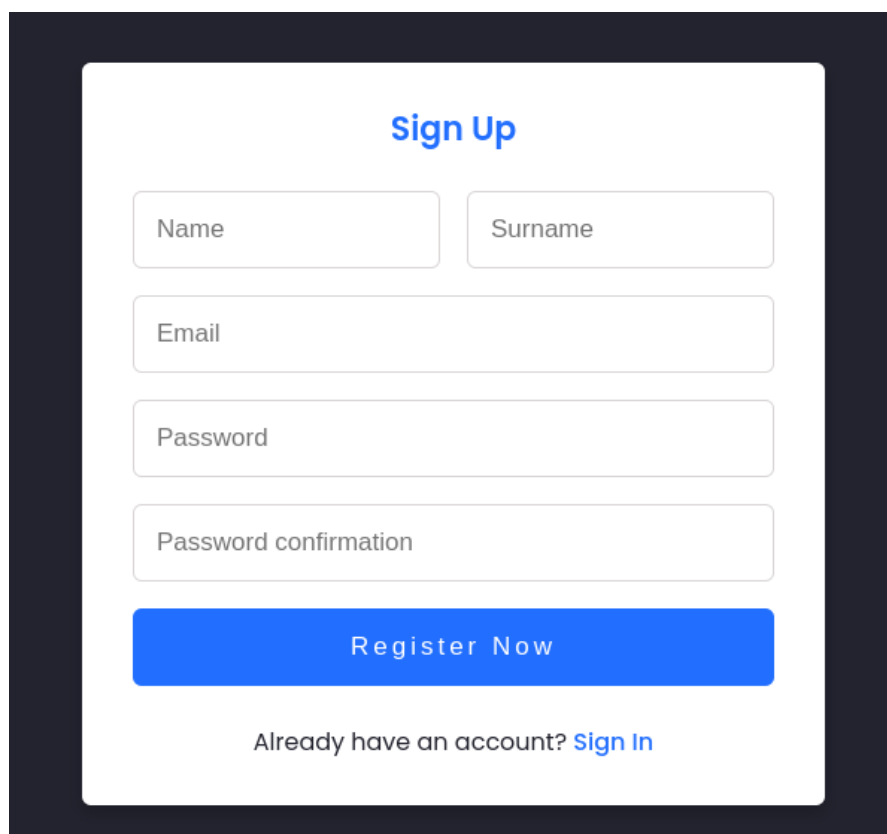
Ниже представлена ключевая часть реализованного интерфейса нашего приложения.

Авторизация пользователей:



The image shows a 'Sign In' form centered on a dark blue background. The form itself is a white rounded rectangle. At the top of the form, the text 'Sign In' is displayed in a blue font. Below this, there are two input fields: the first is labeled 'Enter your email' and the second is labeled 'Enter your password'. Both fields have a light gray border. Underneath the password field is a solid blue button with the text 'Sign In' in white. At the bottom of the form, there is a link that says 'Don't have an account? Sign Up', where 'Sign Up' is a blue hyperlink.

Рисунок 13 — Страница /signin



The image shows a 'Sign Up' form centered on a dark blue background. The form is a white rounded rectangle. At the top, the text 'Sign Up' is in a blue font. Below it are four input fields: 'Name' and 'Surname' are side-by-side, followed by 'Email' and 'Password' stacked vertically. The last field is 'Password confirmation'. All fields have a light gray border. Below these fields is a solid blue button with the text 'Register Now' in white. At the bottom of the form, there is a link that says 'Already have an account? Sign in', where 'Sign in' is a blue hyperlink.

Рисунок 14 — Страница /signup

Реализован поиск товаров по категориям. При нажатии на значок «+» будет осуществляться добавление товара в корзину:

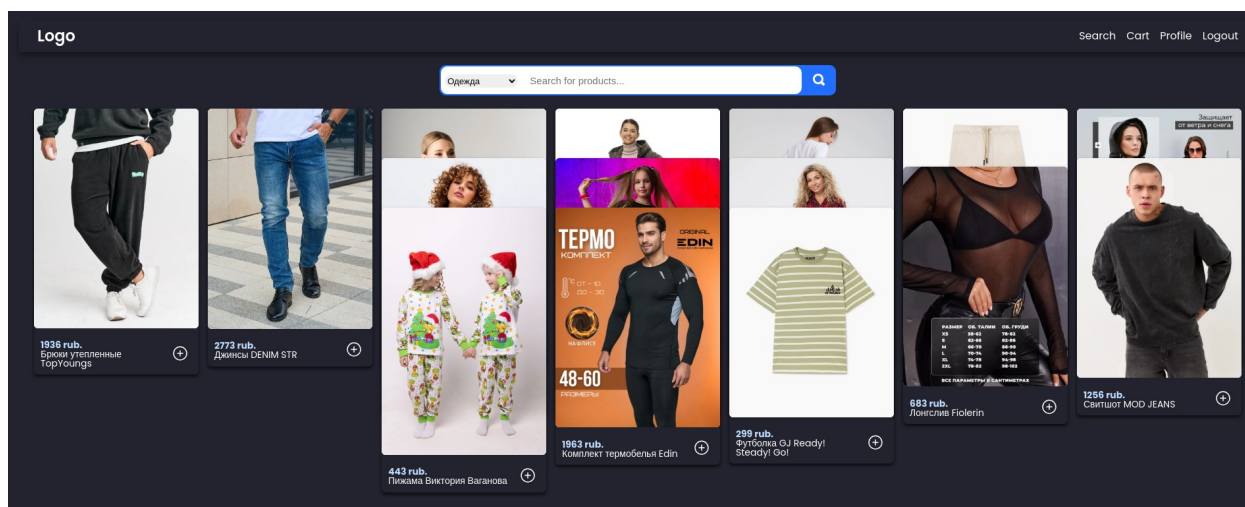


Рисунок 15 — Страница /search

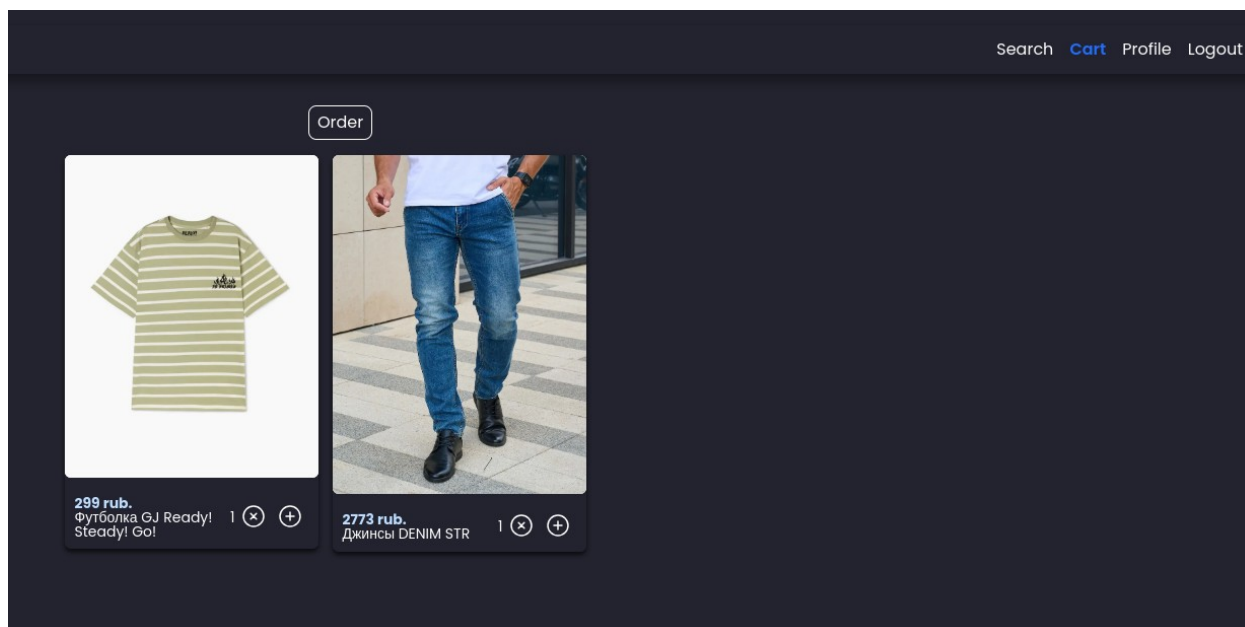


Рисунок 16 — Страница /cart

Рисунок 17 — Страница /order

На странице профиля отображаются все предыдущие заказы пользователя

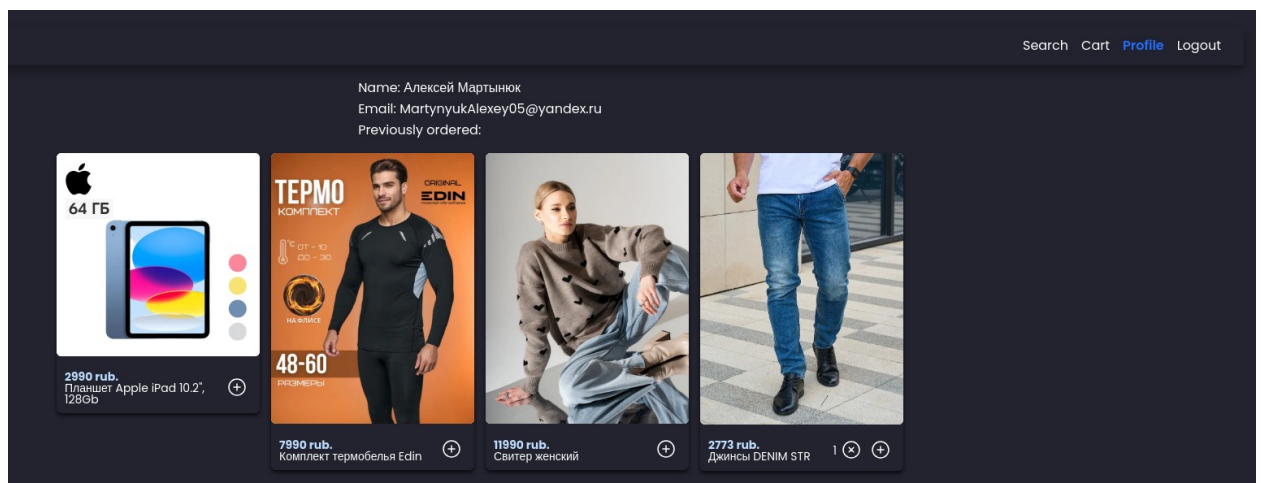


Рисунок 18 — Страница /profile

1.4 Тестирование взаимодействия с системой

Для проверки корректной работы системы создадим заказ и через phpmyadmin(localhost:8080) проверим добавление заказа в базу данных.

Order Form

Total Items: 2
Total Price: 3072 rub.

+12345678901

ул. Тверская, д. 1, кв. 10, Москва, 854239

Meow!

Place Order

Рисунок 19 — Создание заказа

SELECT * FROM `orders`

☐ Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

☐ Show all | Number of rows: 25 | Filter rows: Search this table | Sort by key: None

Extra options

				id	user_id	address_id	status	comment	created_at	updated_at
<input type="checkbox"/>	Edit	Copy	Delete	1	1	1	canceled	NULL	2024-11-03 14:46:43	2024-11-03 15:51:17
<input type="checkbox"/>	Edit	Copy	Delete	2	1	2	completed	NULL	2024-11-13 08:30:41	2024-11-17 13:35:15
<input type="checkbox"/>	Edit	Copy	Delete	3	1	3	progress	NULL	2024-12-02 12:59:09	2024-12-02 13:23:06
<input type="checkbox"/>	Edit	Copy	Delete	4	1	1	progress	Meow!	2024-12-17 20:43:06	2024-12-17 20:43:06

Рисунок 20 — Заказ в бд(id = 4)

1.5 docker-compose.yaml

Для функционирования web-приложения был написан docker-compose файл, который поднимает: серверы для обработки запросов php-fpm и nginx(reverse проху), базу данных MySQL Server и phpmyadmin для доступа к ней.

```
services:
  nginx:
    image: nginx:latest
    container_name: nginx
    volumes:
      - ./public:/usr/share/nginx/html
      - ./nginx.conf:/etc/nginx/nginx.conf:ro
    ports:
      - "80:80"
    depends_on:
      - php-fpm
    healthcheck:
      test: ["CMD", "nginx", "-t"]
      interval: 3s
      retries: 3
  php-fpm:
    build: .
    container_name: php-fpm
    volumes:
      - ./src:/var/www/html
    depends_on:
      - mysql
    healthcheck:
      test: ["CMD", "php", "-v"]
      interval: 3s
      retries: 3
```

Рисунок 21 — nginx и php-fpm

```
mysql:
  image: bitnami/mysql
  container_name: mysql
  restart: always
  volumes:
    - ./migrations:/docker-entrypoint-initdb.d
  ports:
    - 3306:3306
  healthcheck:
    test: ["CMD", "mysqladmin", "ping", "-h", "localhost"]
    interval: 3s
    retries: 3
  environment:
    MYSQL_ROOT_PASSWORD: password
phpmyadmin:
  image: phpmyadmin/phpmyadmin:latest
  container_name: phpmyadmin
  restart: always
  ports:
    - 8080:80
  environment:
    PMA_HOST: mysql
    MYSQL_ROOT_PASSWORD: password
  depends_on:
    - mysql
```

Рисунок 22 — mysql и phpmyadmin

3 WEB-сервер

3.1 Реализация

Для реализации веб-сервера был выбран язык Golang. Создается сервер по принятию подключений. Каждое подключение работает в отдельном потоке. Порт используется дефолтный 8080 или берется из переменных среды.

```
package main

import (
    "context"
    "errors"
    "log"
    "net/http"
    "os"
    "os/signal"
    "syscall"
    "time"
)

Codeium: Refactor | Explain | Generate GoDoc
func main() {
    port := os.Getenv("PORT")
    if port == "" {
        port = "8080"
    }
    server := &http.Server{
        Addr:      ":" + port,
        WriteTimeout: 10 * time.Second,
        ReadTimeout: 10 * time.Second,
        IdleTimeout: 10 * time.Second,
        Handler:    http.FileServer(http.Dir("./")),
    }
    go func() {
        log.Print("Starting serving new connections")
        if err := server.ListenAndServe(); !errors.Is(err, http.ErrServerClosed) {
            log.Print("HTTP server error", "err", err)
            os.Exit(-1)
        }
        log.Print("Stopped serving new connections")
    }()
    sigChan := make(chan os.Signal, 1)
    signal.Notify(sigChan, syscall.SIGINT, syscall.SIGTERM)
    <-sigChan
    shutdownCtx, shutdownRelease := context.WithTimeout(context.Background(), 8*time.Second)
    defer shutdownRelease()
    if err := server.Shutdown(shutdownCtx); err != nil {
        log.Print("Server shutdown error", "err", err)
    }
    log.Print("Server shutdown")
}
```

Рисунок 23 — main.go

3.2 Запуск web-server

Запускаем код из терминала, прерываем процесс сочетанием клавиш `ctrl+c`

```
2024/12/18 00:00:36 Starting serving new connections
^C2024/12/18 00:00:42 Stopped serving new connections
2024/12/18 00:00:42 Server shutdown
```

Рисунок 24 — `go run main.go`

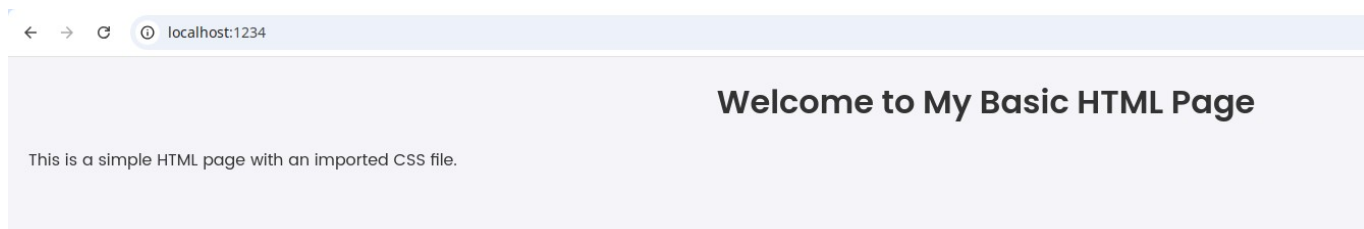


Рисунок 25 — Страница в браузере

ЗАКЛЮЧЕНИЕ

В процессе выполнения лабораторной работы были усовершенствованы навыки программирования на языке php. Кроме того, углублены знания языка html, таблиц стилей css и реляционных баз данных на примере MySQL. Реализован веб-сервер на языке golang, отдающий статические файлы.