Молдавский Государственный Университет

Факультет Математики и Информатики

Департамент Информатики  
  
  
  
  
  
  
**Индивидуальная работа**Веб-приложение “Система заявок на ремонт устройств”   
**0613.4 Informatica**

Выполнил студент II курса  
специальность Информатика  
**Богданов Юрий**

Руководитель, преподаватель  
Департамента Информатики  
**Нартя Никита**

Кишинёв, 2025

Оглавление

[**Цель и основные этапы работы 3**](#_Toc197184532)

[**Теоретическая часть 3**](#_Toc197184533)

[**Формулировка задачи 3**](#_Toc197184534)

[**Основные условия 3**](#_Toc197184535)

[**Функциональные требования к веб-приложению 4**](#_Toc197184536)

[**Требования к безопасности 5**](#_Toc197184537)

[**Архитектура приложения 5**](#_Toc197184538)

[**Документация 6**](#_Toc197184539)

[**Ход работы 6**](#_Toc197184540)

[**Используемая среда 6**](#_Toc197184541)

[**Настройка окружения через `XAMPP` 6**](#_Toc197184542)

[**Инструкции по запуску проекта 7**](#_Toc197184543)

[**Структура базы данных 7**](#_Toc197184544)

[**Структура проекта 10**](#_Toc197184545)

[**Функциональные возможности 12**](#_Toc197184546)

[**Примеры использования и сценарии взаимодействия пользователей с приложением 12**](#_Toc197184547)

[**Ответы на контрольные вопросы 15**](#_Toc197184554)

[**Вывод 16**](#_Toc197184560)

[**Библиография 17**](#_Toc197184561)

# 

# Цель и основные этапы работы

Создание веб-приложения средней сложности с аутентификацией пользователей, формами, ролями и защитой доступа. Реализация функциональности как для обычных пользователей (создание заявок, просмотр статуса), так и для администраторов (управление заявками, слотами времени и пользователями).

# Теоретическая часть

Веб-приложение спроектировано по модульной архитектуре с применением PHP и MySQL. Реализована регистрация, вход, восстановление пароля, создание и фильтрация заявок, разграничение доступа по ролям. Вся серверная логика реализована в PHP с использованием PDO для работы с базой данных.

# Формулировка задачи

Разработать информационную систему, в которой пользователи могут создавать заявки на ремонт устройств, а администраторы — управлять ими. Система должна быть защищена, содержать роли пользователей и администраторов, иметь формы с валидацией и возможность фильтрации заявок.

# Основные условия

1. Разработайте веб-приложение (традиционное веб-приложение, веб-сервис, REST API, мессенджер-бот) средней сложности, содержащее функционал, реализованный на стороне сервера.

2. Для реализации можно использовать любые frontend и backend веб-технологии.

3. Разрешается работа в командах по два человека.

4. Работа должна быть загружена на GitHub, а ссылка прикреплена в Moodle.

5. Индивидуальная работа должна быть представлена преподавателю и коллегам.

## Функциональные требования к веб-приложению

Веб-приложение должно включать следующие компоненты и функциональные возможности:

1. \*\*Аутентификация пользователей\*\*

- Реализуйте механизм регистрации и входа в систему.

- После успешной аутентификации пользователю предоставляется доступ к защищённым разделам сайта.

- Данные аутентификации (\_например\_, логин и пароль) должны храниться безопасным образом с использованием хеширования в базе данных.

- \_Доп. Задание\_. Реализуйте механизм восстановления пароля (\_например\_, через электронную почту).

2. \*\*Общедоступный компонент\*\*

- Раздел приложения, доступный всем пользователям без необходимости авторизации.

- Содержит минимум 2–3 элемента контента, которые формируются динамически с использованием серверных скриптов.

- Данные для отображения должны извлекаться из базы данных.

3. \*\*Формы взаимодействия с пользователем\*\*

В приложении необходимо реализовать как минимум две формы:

- \*\*Форма создания ресурса\*\*

Содержит не менее 5 полей различных типов (текстовые поля, выпадающие списки, переключатели и др.).

Обязательные требования:

- Проверка данных как на стороне клиента, так и на стороне сервера;

- Обработка ошибок и отображение понятных сообщений пользователю.

- \*\*Форма поиска\*\*

Позволяет находить ресурсы по заданным критериям.

4. \*\*Защищённый компонент (только для авторизованных пользователей)\*\*

- Доступен исключительно после входа в систему.

- Для этого компонента необходимо реализовать роль пользователя \*\*«администратор»\*\*.

- Администратор должен иметь доступ к 3–7 дополнительным функциям, включая:

- Создание новых учётных записей с ролью администратора;

- Управление данными в базе данных (просмотр, добавление, редактирование и удаление записей).

## Требования к безопасности

1. Валидируйте все данные, введенные в формы, чтобы предотвратить внедрение вредоносного кода.

2. Доступ к закрытым частям приложения должен быть защищён аутентификацией (например, пароль).

3. Пароли должны храниться в базе данных с использованием безопасных хэш-функций.

4. Используйте сессии и переменные сессии (или токены) для управления доступом, чтобы предотвратить обход аутентификации.

## Архитектура приложения

1. Постройте приложение на модульной архитектуре для удобства расширения и поддержки.

2. Опционально используйте архитектуру MVC (Model-View-Controller), чтобы улучшить структуру приложения.

## Документация

1. \*\*Код\*\*:

- Оформляйте код с использованием PHPDoc.

- Каждая функция и метод должны содержать описание их входных параметров, выходных данных и функционала.

- Комментарии должны быть понятными и информативными.

- Не допускайте избыточных комментариев, которые не добавляют ценности кода.

- Используйте понятные и описательные имена переменных, функций и классов.

## Ход работы

* **\*\*Система заявок на ремонт устройств\*\*** — это веб-приложение, позволяющее пользователям регистрироваться, отправлять заявки на ремонт, отслеживать их статус, а администраторам — управлять заявками и расписанием. Проект реализован на PHP с использованием MySQL и развернут в среде XAMPP.

### Используемая среда

- СУБД: \*\*MySQL\*\* (через интерфейс \*\*phpMyAdmin\*\*, установленный в составе \*\*XAMPP\*\*)

- Веб-сервер: \*\*Apache\*\* (через \*\*XAMPP\*\*)

- Язык: \*\*PHP 8.2\*\*

- Интерфейс управления базой: \*\*phpMyAdmin\*\* (localhost)

### Настройка окружения через `XAMPP`

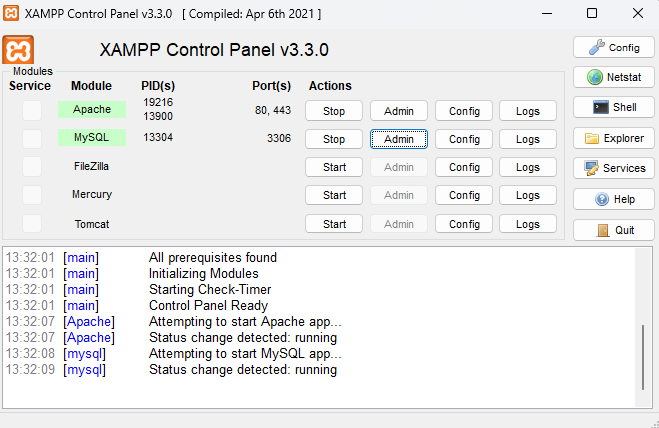
Для локальной разработки использован программный комплекс `XAMPP`, включающий в себя:

1. `Apache` — для запуска PHP-приложения на локальном сервере;

2. `MySQL` — сервер баз данных, работающий совместимо с MariaDB;

3. `phpMyAdmin` — визуальный интерфейс для администрирования базы данных.

На панели управления `XAMP` были запущены модули:



* Благодаря `XAMPP`, разработка происходила в локальной среде по адресу [http://localhost/repairTicketSystem/public/index.php](http://localhost/repairTicketSystem/public/index.php)

### Инструкции по запуску проекта

1. Установите `XAMPP` и запустите `Apache` и `MySQL`.

2. Клонируйте репозиторий проекта в папку `C:\xampp\htdocs`:

3. мпортируйте файл базы данных `init.sql` из папки `sql` в `phpMyAdmin`.

4. Перейдите по ссылке

[http://localhost/repairTicketSystem/public/index.php](http://localhost/repairTicketSystem/public/index.php).

5. Зарегистрируйтесь или войдите в систему для использования функционала.

### Структура базы данных

#### Для данного проекта была создана база данных `repair\_system`, структура которой описана в SQL-скрипте `sql/init.sql`. Этот файл содержит команды создания всех необходимых таблиц и связей между ними

#### Таблица `users`

CREATE TABLE users (

id INT AUTO\_INCREMENT PRIMARY KEY,

username VARCHAR(50) NOT NULL UNIQUE,

email VARCHAR(100) NOT NULL UNIQUE,

password VARCHAR(255) NOT NULL,

role ENUM('user', 'admin') DEFAULT 'user'

);

**Объяснение:**

Содержит данные зарегистрированных пользователей: `логин`, `email`, `пароль` (в хешированном виде), а также роль пользователя (`user` или `admin`).

- `username` — имя пользователя, уникальное.

- `email` — email-адрес, также уникальный.

- `password` — хэш пароля.

- `role` — роль в системе (пользователь или администратор).

#### Таблица `categories`

CREATE TABLE categories (

id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(100) NOT NULL,

description TEXT

);

**Объяснение:**

Список доступных категорий (услуг). Используется пользователями при создании заявки.

- `name` — название услуги.

- `description` — описание услуги.

#### Таблица `devices`

CREATE TABLE devices (

id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(50) NOT NULL

)

**Объяснение:**

Содержит список возможных типов устройств (например, ноутбук, телефон и т.д.).

- `name` — название устройства (например, ноутбук, телефон).

#### Таблица `time\_slots`

CREATE TABLE time\_slots (

id INT AUTO\_INCREMENT PRIMARY KEY,

slot\_time DATETIME NOT NULL,

is\_booked BOOLEAN DEFAULT 0

);

**Объяснение:**

Хранит доступные слоты времени для записи на ремонт. Администратор может добавлять слоты, и каждый слот может быть помечен как занятый.

- `slot\_time` — дата и время записи.

- `is\_booked` — статус (занят/свободен)

#### Таблица `requests`

CREATE TABLE requests (

id INT AUTO\_INCREMENT PRIMARY KEY,

user\_id INT NOT NULL,

category\_id INT NOT NULL,

device\_id INT NOT NULL,

problem\_description TEXT NOT NULL,

urgency ENUM('низкая', 'средняя', 'высокая') NOT NULL,

time\_slot\_id INT NOT NULL,

status ENUM('ожидание', 'подтверждено', 'отклонено') DEFAULT 'ожидание',

created\_at DATETIME DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (user\_id) REFERENCES users(id),

FOREIGN KEY (category\_id) REFERENCES categories(id),

FOREIGN KEY (device\_id) REFERENCES devices(id),

FOREIGN KEY (time\_slot\_id) REFERENCES time\_slots(id)

);

**Объяснение:**

Основная таблица, где хранится информация о заявках пользователей. Каждая заявка связана с пользователем, категорией, устройством и выбранным временем. Также фиксируется срочность, статус обработки и дата создания.

### Структура проекта

repairTicketSystem/

├── README.md # Описание проекта и условия выполнения работы

├── public/

│ ├── index.php # Главная точка входа

│ └── assets/

│ ├── style.css # Стили оформления интерфейса

│ └── images/

│ └── background.png # Фоновое изображение

├── sql/

│ └── init.sql # SQL-скрипт для создания структуры базы данных

├── src/

│ ├── handlers/

│ │ ├── add\_slot\_handler.php # Обработка формы добавления слота (админ)

│ │ ├── admin\_requests\_handler.php # Получение всех заявок для админ-панели

│ │ ├── create\_request\_handler.php # Обработка создания заявки пользователем

│ │ ├── login\_handler.php # Обработка входа пользователя

│ │ ├── logout\_handler.php # Завершение сессии пользователя

│ │ ├── password\_reset\_handler.php # Сброс пароля по коду

│ │ ├── password\_reset\_request\_handler.php # Отправка кода сброса пароля

│ │ └── register\_handler.php # Обработка регистрации нового пользователя

│ ├── helpers/

│ │ ├── db.php # Функция подключения к базе данных через PDO

│ │ ├── router.php # Простая маршрутизация страниц

│ │ └── view.php # Рендер шаблонов с layout

│ ├── models/

│ │ ├── CategoryModel.php # Получение всех категорий из базы данных

│ │ ├── DeviceModel.php # Получение всех типов устройств

│ │ └── SlotModel.php # Получение всех свободных слотов

│ └── templates/

│ ├── add\_slot.php # Форма добавления слота (для админа)

│ ├── admin\_panel.php # Панель управления заявками (админ)

│ ├── create\_request.php # Форма создания заявки на ремонт

│ ├── dashboard.php # Панель пользователя (просмотр заявок)

│ ├── home.php # Главная страница (категории)

│ ├── layout.php # Базовый макет страницы с меню и footer

│ ├── login.php # Страница входа

│ ├── password\_reset.php # Страница сброса пароля по коду

│ ├── password\_reset\_request.php # Страница запроса кода сброса пароля

│ ├── register.php # Страница регистрации нового пользователя

│ └── slots.php # Список всех слотов (с удалением)

### Функциональные возможности

Веб-приложение предоставляет чётко разграниченные функции в зависимости от роли пользователя. Незарегистрированный посетитель может просматривать главную страницу и список доступных услуг — это открытая часть системы. После регистрации и входа пользователю становятся доступны формы для создания заявок на ремонт. Пользователь может указать устройство, выбрать категорию ремонта, описать проблему, указать срочность и выбрать доступный временной слот. Все эти данные обрабатываются, валидируются и сохраняются в базу.

После отправки заявки пользователь может перейти в личный кабинет, где отображаются все его обращения с указанием их статуса. Заявки можно фильтровать по статусу: \*\*"ожидание"\*\*, \*\*"подтверждено"\*\* или \*\*"отклонено"\*\*. Это позволяет отслеживать прогресс по каждому обращению.

Администратор, войдя в систему, получает доступ к административной панели. Он может просматривать все пользовательские заявки, изменять их статус (\*\*подтвердить\*\*, \*\*отклонить\*\*), а также удалять заявки при необходимости. При удалении автоматически освобождается соответствующий слот времени. Администратор может также добавлять новые временные слоты через отдельную форму. Дополнительно он имеет доступ к просмотру списка всех доступных слотов и их статусов (\*\*занят\*\* или \*\*свободен\*\*).

Функциональность по восстановлению пароля позволяет пользователю, забывшему свой пароль, запросить код восстановления на указанный email. После получения кода он может установить новый пароль.

Таким образом, система охватывает полный цикл работы с заявками на ремонт: от регистрации и создания заявки до её обработки и администрирования.

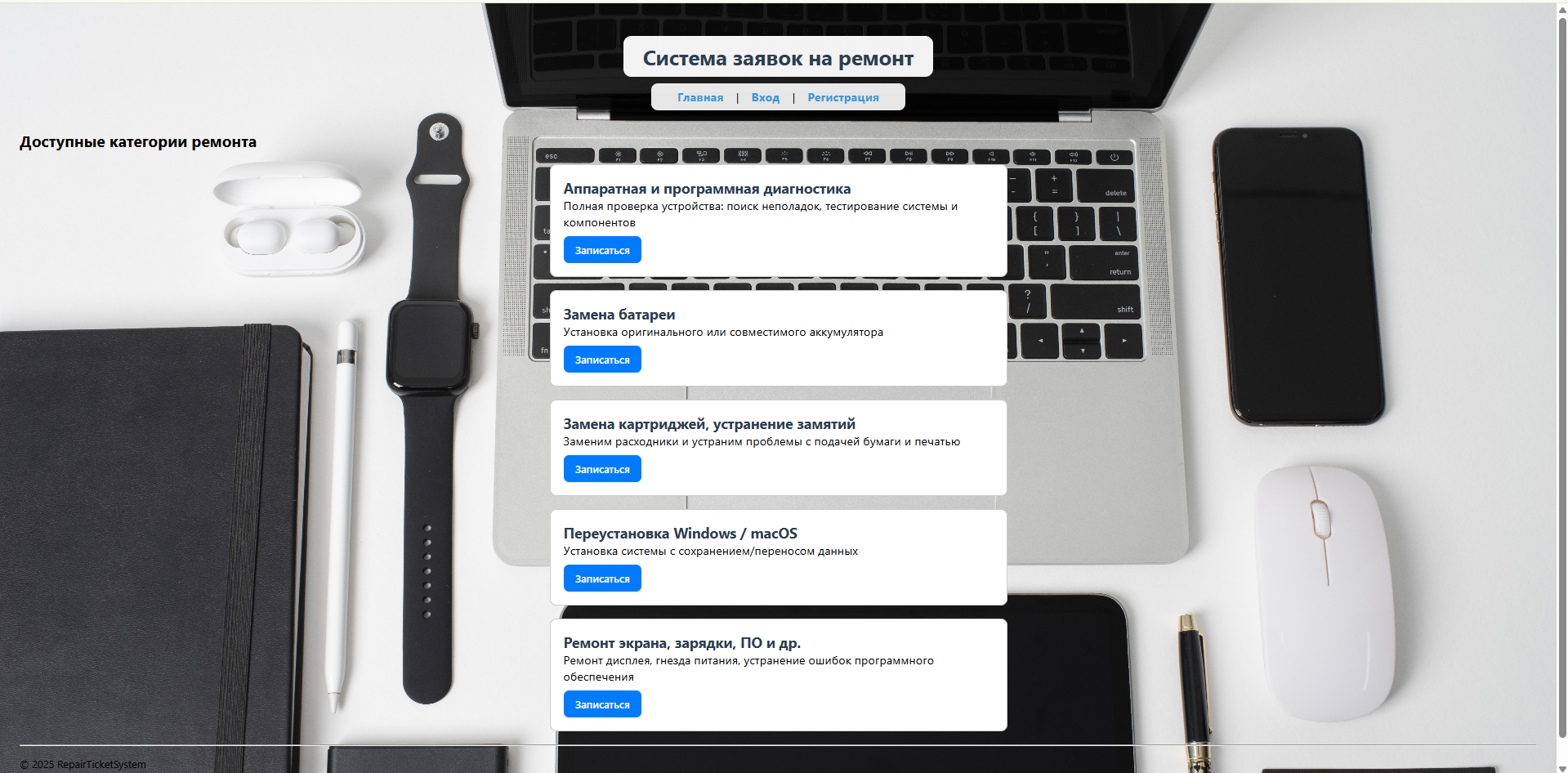
### Примеры использования и сценарии взаимодействия пользователей с приложением

В приложении предусмотрены два основных типа пользователей: `обычный` пользователь и `администратор`. Каждый тип взаимодействует с системой по-своему.

## - \*\*Гость (неавторизованный пользователь)\*\*

При входе на главную страницу гость видит доступные категории услуг (данные берутся из БД):

$categories = CategoryModel::getAll();



Пользователь может нажать `«Записаться»`, но для создания заявки потребуется авторизация:

if (!isset($\_SESSION['user'])) {

echo 'Вы должны войти, чтобы оставить заявку.';

return;

}

## https://i.imgur.com/cfo6lNf.png

## - \*\*Регистрация и вход\*\*

Пользователь заполняет форму регистрации: `логин`, `email`,`пароль`, `подтверждение пароля`. Данные валидируются и сохраняются в БД.

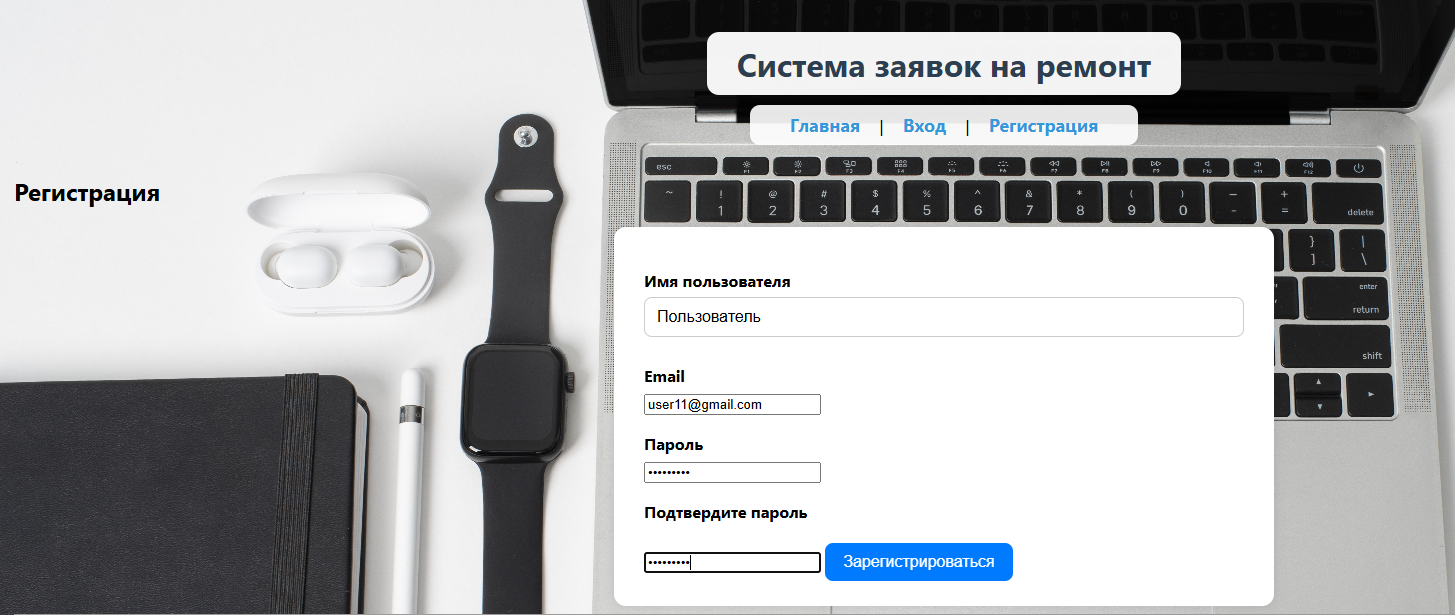
if ($\_SERVER['REQUEST\_METHOD'] === 'POST') {

// Проверка совпадения паролей и хеширование

$hash = password\_hash($\_POST['password'], PASSWORD\_DEFAULT);

$stmt = $pdo->prepare("INSERT INTO users (...) VALUES (...)");

}

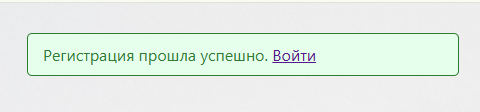


**После регистрации возможен вход в систему:**

if (password\_verify($enteredPassword, $user['password'])) {

$\_SESSION['user'] = $user;

}



## https://i.imgur.com/z4OUFC3.png

## - \*\*Создание заявки\*\*

Авторизованный пользователь переходит к форме создания заявки:

Поля: `категория`, `устройство`, `описание`, `срочность`, `слот времени`.

if ($\_SERVER['REQUEST\_METHOD'] === 'POST') {

$stmt = $pdo->prepare("INSERT INTO requests (...) VALUES (...)");

$stmt->execute([...]);

}

## C:\Users\Asus\OneDrive\Изображения\Снимки экрана\Снимок экрана 2025-05-03 153114.png

## https://i.imgur.com/PQswaTB.png

## - \*\*Просмотр и фильтрация заявок\*\*

На странице "Мои заявки" пользователь может просматривать все отправленные заявки и фильтровать их по статусу:

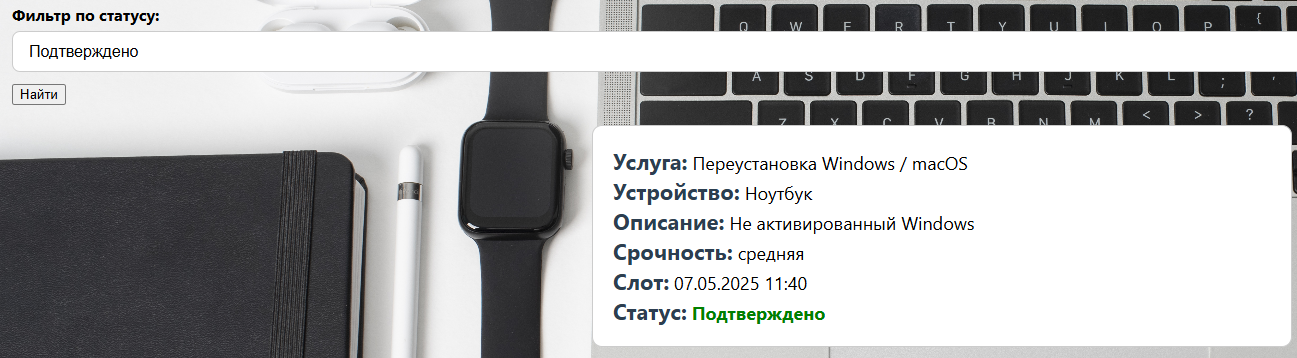
if ($statusFilter) {

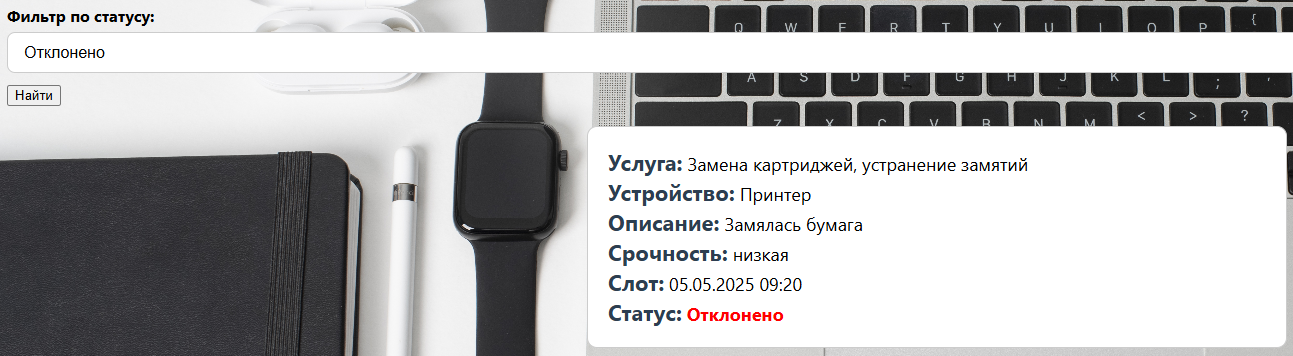
$stmt = $pdo->prepare("SELECT ... WHERE status = ?");

}

## C:\Users\Asus\OneDrive\Изображения\Снимки экрана\Снимок экрана 2025-05-03 154231.png

## C:\Users\Asus\OneDrive\Изображения\Снимки экрана\Снимок экрана 2025-05-03 154305.png





## - \*\*Восстановление пароля\*\*

Если пользователь забыл пароль, он может:

- Запросить код восстановления;

- Ввести email и новый пароль.

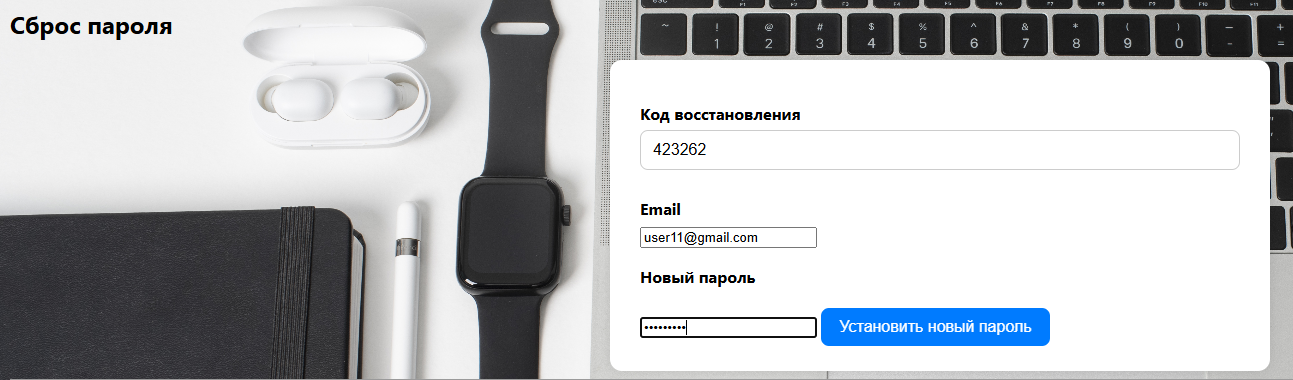
if ($\_POST['reset\_code'] === $storedCode) {

$stmt = $pdo->prepare("UPDATE users SET password = ? WHERE email = ?");

}

## C:\Users\Asus\OneDrive\Изображения\Снимки экрана\Снимок экрана 2025-05-03 153902.png

## C:\Users\Asus\OneDrive\Изображения\Снимки экрана\Снимок экрана 2025-05-03 154055.png



## - \*\*`Администратор`: управление системой\*\*

После входа администратор получает доступ к админ-панели:

Он может:

- Просматривать заявки;

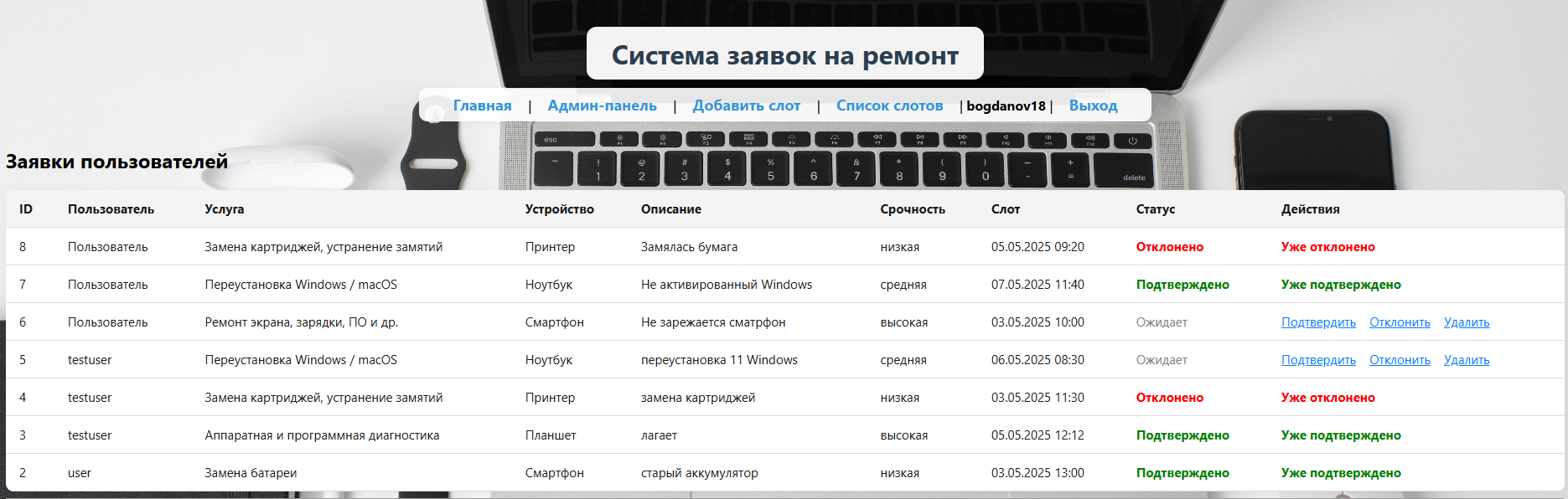
- Менять их статус;

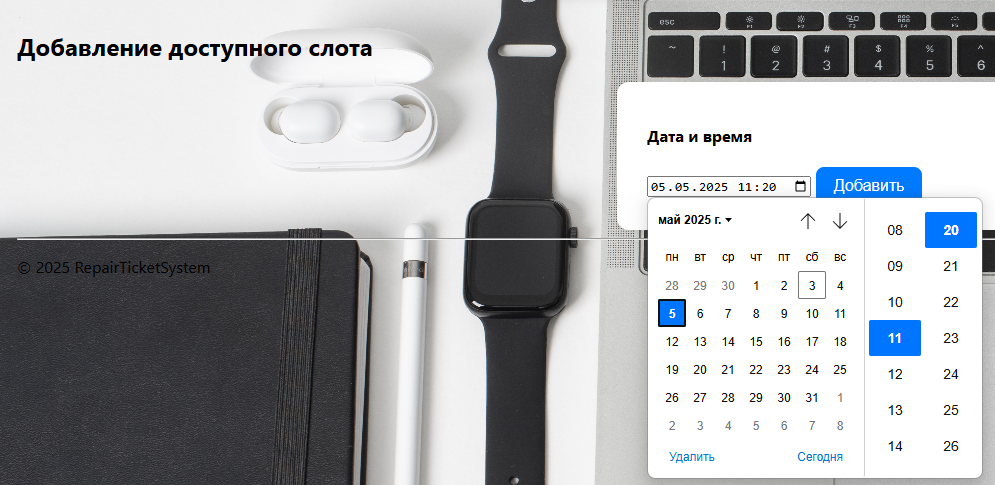
- Удалять заявки (освобождая слоты);

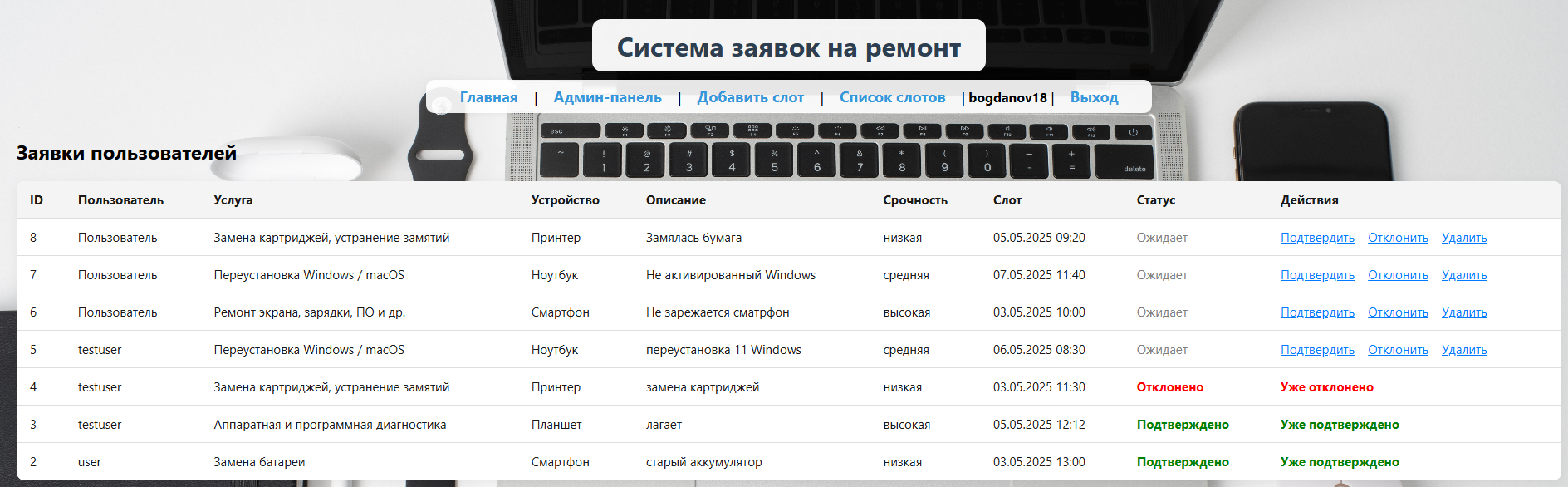
- Добавлять слоты для записи.

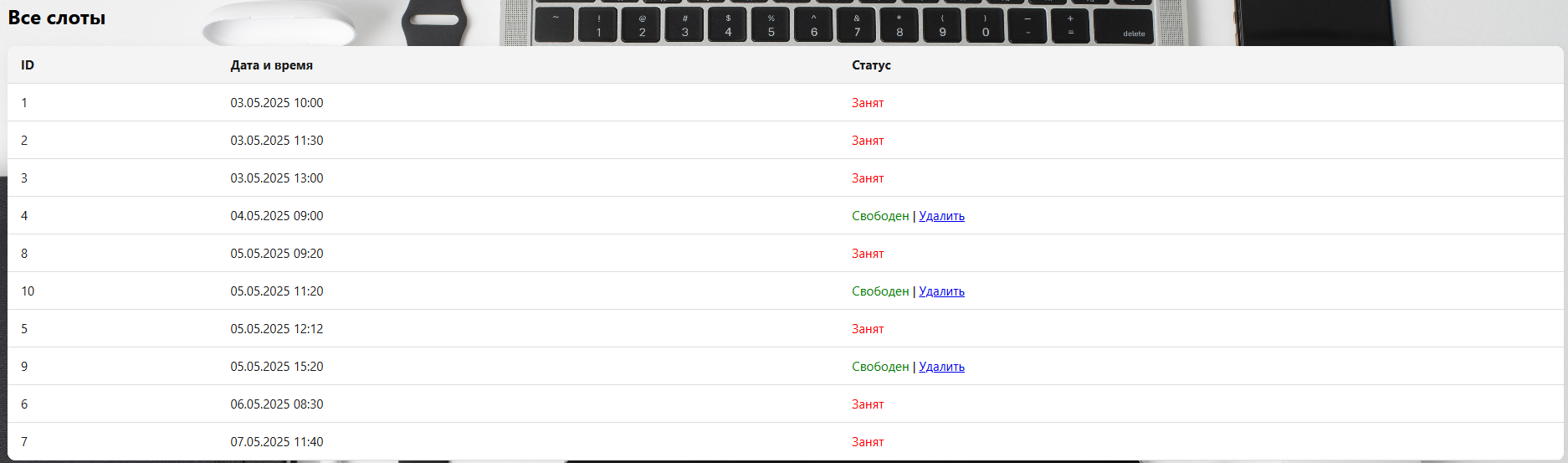
if ($\_SESSION['user']['role'] === 'admin') {

}









## Ответы на контрольные вопросы

### Какие технологии использовались при разработке проекта?

При создании проекта использовался язык программирования PHP версии 8.2 для обработки серверной логики. В качестве базы данных применялась MySQL, управляемая через графический интерфейс phpMyAdmin. Для запуска веб-сервера использовался Apache, входящий в состав пакета XAMPP. Интерфейс реализован на HTML и CSS. Структура проекта построена модульно: вся маршрутизация выполняется через единый файл `index.php`, а логика обработки разбита на обработчики (`handlers`), модели и шаблоны.

### Как реализована аутентификация пользователей?

Аутентификация обеспечивается через формы регистрации и входа. При регистрации пароль пользователя шифруется с помощью функции `password\_hash()` и сохраняется в базу данных. При последующем входе введённый пароль проверяется с использованием `password\_verify()`. После успешной проверки в сессию сохраняются данные пользователя, что позволяет отслеживать его авторизацию на протяжении всей сессии и ограничивать доступ к защищённым разделам.

### Какие меры безопасности реализованы в приложении?

Система защищена на нескольких уровнях. Все пользовательские данные проходят проверку на стороне сервера и клиента, что предотвращает внедрение вредоносного кода (XSS, SQL-инъекций и т.д.). Пароли пользователей хранятся только в зашифрованном виде. Доступ к административным функциям возможен только после проверки роли, сохранённой в сессии. Кроме того, при обращении к защищённым разделам всегда выполняется проверка авторизации.

### Какие роли пользователей предусмотрены и как они разграничивают доступ?

В приложении предусмотрены две роли: обычный пользователь и администратор. Обычный пользователь может зарегистрироваться, войти в систему, создать заявку на ремонт и отслеживать её статус. Администратор имеет расширенные права: он видит все заявки всех пользователей, может изменять их статус (подтверждать, отклонять), удалять их, а также управлять временными слотами (добавлять или освобождать).

### Как реализован механизм восстановления пароля?

Пользователь, забывший пароль, может запросить восстановление, указав свой email. Система отправляет на почту код подтверждения, который затем необходимо ввести в отдельной форме. После успешной проверки кода пользователь может задать новый пароль. Вся информация обновляется в базе данных, и пользователь получает доступ к системе с новым паролем. Этот процесс обеспечивает безопасность и предотвращает несанкционированное восстановление доступа

## Вывод

В ходе выполнения индивидуального проекта была успешно разработана и протестирована веб-система для управления заявками на ремонт устройств. Проект охватывает полный жизненный цикл обработки заявок: от регистрации пользователя и подачи обращения до его обработки администратором. Приложение реализовано с использованием PHP, базы данных MySQL и веб-сервера Apache в среде XAMPP.

Особое внимание было уделено безопасности: реализована защита с помощью сессий, хеширование паролей, валидация данных на сервере. В проекте реализована ролевая модель доступа: пользователь и администратор имеют разные уровни возможностей.

Все функции были реализованы строго в соответствии с техническими требованиями: регистрация и вход, создание и фильтрация заявок, админ-панель, восстановление пароля, работа с временными слотами и шаблонная архитектура. Проект обладает удобной и расширяемой структурой, что позволяет легко вносить улучшения в будущем.

Система была успешно развернута локально и прошла тестирование со стороны разных ролей. Таким образом, поставленные цели и задачи были полностью достигнуты.

## Библиография

1. [Официальная документация PHP (v8.2)](https://www.php.net/manual/ru/) — описание синтаксиса, функций, работы с сессиями, формами и безопасностью.

2. [Работа с базой данных через PDO](https://www.php.net/manual/ru/book.pdo.php) — руководство по безопасному подключению к БД и выполнению запросов.

3. [XAMPP — локальный сервер для PHP и MySQL](https://www.apachefriends.org/ru/index.html) — официальный сайт программного комплекса.

4. [Документация по MySQL и SQL](https://dev.mysql.com/doc/) — справочник по структурам таблиц, типам данных, командам SQL.

5. [Руководство по HTML и формам на MDN](https://developer.mozilla.org/ru/docs/Web/HTML/Element/Form) — для создания форм регистрации, авторизации и создания заявок.

6. [Безопасность в веб-приложениях: валидация, хэширование, XSS и SQL-инъекции](https://owasp.org/www-project-top-ten/) — рекомендации от OWASP по защите пользовательских данных.

7. [PHP Sessions и авторизация](https://www.php.net/manual/ru/book.session.php) — для управления входом, ролями и доступом к страницам.

8. [PHPDoc — документирование кода PHP](https://docs.phpdoc.org/) — формат описания функций, параметров и классов.

**Ссылка на репозиторий GitHub**: