**МИНОБРНАУКИ РОССИИ**

федеральное государственное автономное образовательное учреждение

высшего образования

«Санкт-Петербургский политехнический университет Петра Великого»

(ФГАОУ ВО «СПбПУ»)

**Институт среднего профессионального образования**

**ОТЧЕТ**

**по учебной практике УП.02.01 (по профилю специальности)**

по профессиональному модулю ПМ.02 «Осуществление интеграции программных модулей»

(код и наименование)

Специальность09.02.07 Информационные системы и программирование

(код и наименование специальности)

Студент(ка) 4 курса 42919/7 группы

Орлов Александр Александрович

(Фамилия, имя, отчество)

Место прохождения практики: УВЦ, пр. Энгельса, 23

(наименование и адрес организации)

Период прохождения практики

с «28» октября 2024 г. по «09» ноября 2024 г.

Руководитель практики Хисамутдинова А.С.

(подпись) (расшифровка подписи)

Итоговая оценка по практике \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Санкт-Петербург

2024

**ЗАДАНИЕ**

**на учебную практику (по профилю специальности)**

по профессиональному модулю ПМ.02 «Осуществление интеграции программных модулей»

(код и наименование)

Специальность09.02.07 Информационные системы и программирование

(код и наименование специальности)

Студент(ка) 4 курса 42919/7 группы

Орлов Александр Александрович

(Фамилия, имя, отчество)

Место прохождения практики: УВЦ, пр. Энгельса, 23

(наименование и адрес организации)

Период прохождения практики

с «28» октября 2024 г. по «09» ноября 2024 г.

**Виды работ, обязательные для выполнения** *(переносится из программы, соответствующего ПМ):*

* Участие в выработке требований к программному обеспечению;
* Стадии проектирования программного обеспечения. Проектирование UML-диаграмм;
* Конструирование пользовательского интерфейса. Разработка модулей программного обеспечения;
* Разработка рабочего проекта и технологической документации.

**Индивидуальное задание: ВАРИАНТ 12**

Задание выдал «28» октября 2024 г. \_\_\_\_\_\_\_\_\_\_\_\_\_ Хисамутдинова А.С.

(подпись) (Ф.И.О.)

Задание получил «28» октября 2024 г. \_\_\_\_\_\_\_\_\_\_\_\_ Орлов А.А. (подпись) (Ф.И.О.)

**МИНОБРНАУКИ РОССИИ**

федеральное государственное автономное образовательное учреждение

высшего образования

«Санкт-Петербургский политехнический университет Петра Великого»

(ФГАОУ ВО «СПбПУ»)

**Институт среднего профессионального образования**

**ДНЕВНИК**

**прохождения учебной практики УП.02.01**

**(по профилю специальности)**

по профессиональному модулю ПМ.02 «Осуществление интеграции программных модулей»

(код и наименование)

Специальность09.02.07 Информационные системы и программирование

(код и наименование специальности)

Студент(ка) 4 курса 42919/7 группы

Орлов Александр Александрович

(Фамилия, имя, отчество)

Место прохождения практики: УВЦ, пр. Энгельса, 23

(наименование и адрес организации)

Период прохождения практики

с «28» октября 2024 г. по «09» ноября 2024 г.

Руководитель практики Хисамутдинова А.С.

(подпись) (расшифровка подписи)

Итоговая оценка по практике

Санкт-Петербург

2024

**Содержание дневника**

|  |  |  |
| --- | --- | --- |
| **Дата** | **Виды выполненных работ и заданий по программе практики** | **Подпись руководителя практики** |
| **1** | **2** | **3** |
| 28.10.2024 | Анализ предметной области. Анализ методов решения. Анализ и выбор средств. |  |
| 29.10.2024 | Анализ и выбор средств. Исследовательские работы. Разработка технического задания. |  |
| 30.10.2024 | UML. Проектирование диаграммы вариантов использования и последовательности. UML. Проектирование диаграммы вариантов активности и состояния. |  |
| 31.10.2024 | Моделирование структуры ПО. Проектирование инфологической и даталогической модели данных. |  |
| 01.11.2024 | Проектирование интерфейса пользователя. |  |
| 02.11.2024 | Разработка схемы алгоритма программного продукта. |  |
| 05.11.2024 | Разработка модели данных. Разработка программного продукта. Создание функциональности ПО. |  |
| 06.11.2024 | Модульное тестирование. Создание тестовых случаев. |  |
| 07.11.2024 | Создание тестовых случаев. Отладка программных модулей. |  |
| 08.11.2024 | Разработка документа «Руководство пользователя» в соответствии с ЕСПД. Разработка документа «Программа и методика испытаний» в соответствии с ЕСПД. |  |
| 09.11.2024 | Подготовка отчета. |  |

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 6](#_Toc181866098)

[Анализ предметной области. Анализ методов решения. Анализ и выбор средств. 7](#_Toc181866099)

[Анализ и выбор средств. Исследовательские работы. Разработка технического задания. 9](#_Toc181866100)

[UML. Проектирование диаграммы вариантов использования и последовательности. UML. Проектирование диаграммы вариантов активности и состояния. 11](#_Toc181866101)

[Моделирование структуры ПО. Проектирование инфологической и даталогической модели данных. 14](#_Toc181866102)

[Проектирование интерфейса пользователя. 22](#_Toc181866103)

[Разработка модели данных. Разработка программного продукта. Создание функциональности ПО. 29](#_Toc181866104)

[Модульное тестирование. Создание тестовых случаев. 33](#_Toc181866105)

[Создание тестовых случаев. Отладка программных модулей. 38](#_Toc181866106)

[ЗАКЛЮЧЕНИЕ 47](#_Toc181866107)

ВВЕДЕНИЕ

Создание ТЗ является важной частью cоздания информационного продукта. Оно служит основой для последующей разработки и должно служить опорой для создания качественного продукта для заказчика. Моделирование данных, в свою очередь, помогает визуализировать структуру и логику системы, обеспечив возможность установки взаимосвязей между сущностями и формирования базы данных, а также предоставить заказчику макет приложения для оценки ещё до основной разработки.

Анализ предметной области. Анализ методов решения. Анализ и выбор средств.

Основной целью данного модуля является страхование банковских вкладов. Модуль позволяет клиентам страховать свои вклады и быстро производить выплаты вкладчикам из независимых финансовых источников в случае прекращения деятельности банка.

Разработка программного модуля для страхования вкладов – это процесс создания программного продукта, который позволяет банкам эффективно отслеживать страхование вкладов и производить выплаты вкладчикам.

Задача разработки такого программного модуля состоит в создании удобного и эффективного инструмента для страхования вкладов, быстрых выплат, что помогает минимизировать время простоя оборудования и обеспечивает бесперебойную работу в банковской сфере.

Программный модуль позволяет автосервису повысить эффективность и скорость выплат, сократить время обработки заявок, оптимизировать использование ресурсов и повысить удовлетворенность клиентов.

Основные функции и возможности модуля включают:

1. Заявка на страхование: это информация, предоставленная клиентом, которая позволяет идентифицировать его от остальных клиентов. Клиент оставляет новую заявку. Заявка может содержать данные о сумме вклада, личную информацию (ФИО клиента, номер телефона, паспортные данные, банковские реквизиты).

2. Регистрация заявки: этот процесс включает приём и регистрацию заявки оператором в системе учёта. Важными аспектами регистрации являются присвоение уникального идентификатора заявке, сохранение информации о заявке.

3. Обработка заявки: процесс, включающий анализ заявки, определение процента страхования.

4. Исполнение заявки: фактическая выплата вкладчику. На этом этапе клиенту перечисляется процент от его вклада, исполнитель вносит необходимые изменения.

5. Отчётность и информирование: важной составляющей учёта заявок на ремонт является фиксация и отчёт о выполненной работе. После завершения услуги страхования вклада, исполнитель должен предоставить отчёт, содержащий следующую информация: кому выдано, сколько выдано, когда выдано.

Анализ и выбор средств. Исследовательские работы. Разработка технического задания.

1. Общие сведения

1.1. Наименование проекта: Разработка программного модуля для проведения праздников

1.2. Заказчик: Кащаев Никита Витальевич.

1.3. Исполнитель: Орлов Александр Александрович.

2. Функциональные требования

2.1. Возможность добавления заявок в базу данных с указанием следующих параметров:

- Номер заявки;

- Дата добавления;

- ФИО Клиента;

- Номер телефона клиента;

- Количество гостей;

- Место проведения мероприятия;

- Вид мероприятия;

- Статус заявки (в oпроцессe, завершена).

- Приоритетность заказа;

2.2. Возможность редактирования заявок:

- Изменение места проведения, количества гостей, контактной информации клиента;

- Изменение ответственного за выполнение работ.

2.3. Возможность отслеживания статуса заявки:

- Отображение списка заявок;

2.4. Возможность назначения ответственных за выполнение работ:

- Добавление исполнителя к заявке;

- Отслеживание состояния работы;

2.5. Расчет статистики работы отдела обслуживания:

- Количество выполненных заявок;

- Среднее время выполнения заявки;

3. Нефункциональные требования

3.1. Кроссплатформенность:

- Поддержка работы на ОС семейства Windows.

3.2. Безопасность:

- Логин и пароль для доступа к приложению;

- Доступ к данным должен быть ограничен в зависимости от роли пользователя.

3.3. Удобство использования:

- Простой и интуитивный интерфейс;

3.4. Производительность:

- Приложение должно иметь быстрый доступ к данным;

- Минимальное время отклика на запросы пользователя.

4. Требования к реализации

4.1. Язык программирования: PHP

4.2. СУБД: MySQL

5. Требования к документации

5.1. Техническое задание на разработку программного модуля.

6. Руководство по стилю

6.1. Шрифт: Times New Roman, 14пт.

6.2. Цветовая схема:

Фон: светлый кремовый (#FFF9E6)

Основной текст: черный (#000000)

Заголовки: ярко-красный (#D32F2F)

Кнопки: оранжевый (#FFA000) с белым текстом (#FFFFFF)

Акценты: яркий зеленый (#66BB6A) для элементов взаимодействия

UML. Проектирование диаграммы вариантов использования и последовательности. UML. Проектирование диаграммы вариантов активности и состояния.

Чтобы лучше понять устройство будущего приложения сделаем UML диаграммы прецендентов (Рисунок 1), последовательности (Рисунок 2), поведенческой (Рисунок 3).

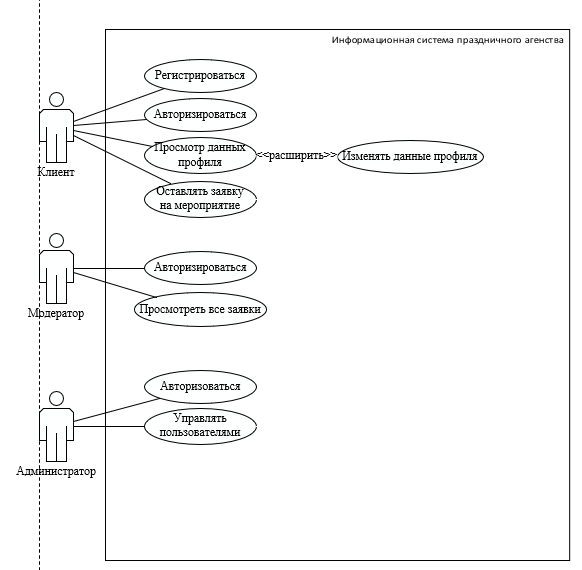


Рисунок 1 – Диаграмма прецендентов

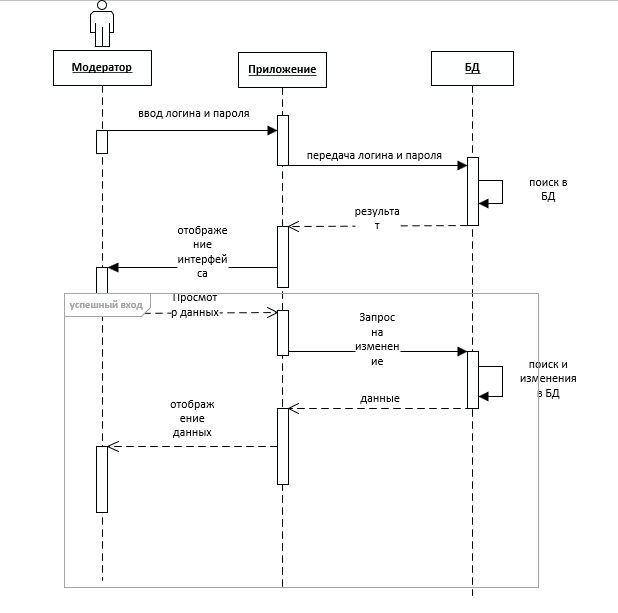


Рисунок 2 – Диаграмма последовательности

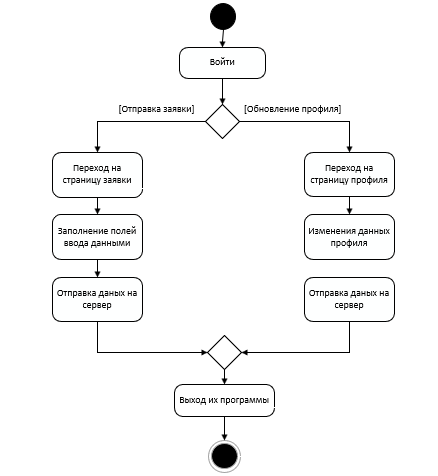


Рисунок 3 – Диаграмма поведенческая

Моделирование структуры ПО. Проектирование инфологической и даталогической модели данных.

Для лучшего проектирования базы данных создадим даталогическую (Рисунок 4) модель и инфологическую модели.

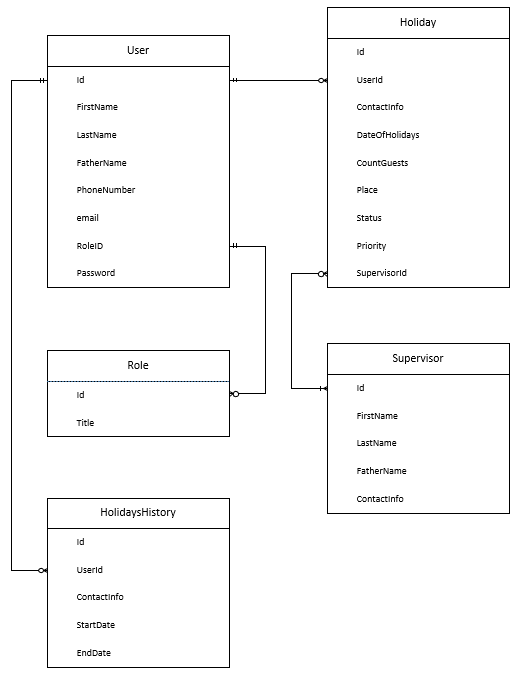


Рисунок 4 – Датологическая модель данных

Для лучшего раскрытия полей в базе данных нам потребуется словарь данных – инфологическая модель, он представлен ниже.

Словарь данных.

User

|  |  |
| --- | --- |
| Id (PK) | Int |
| FirstName | Varchar(50) |
| SecondName | Varchar(50) |
| FatherName | Varchar(50) |
| PhoneNumber | Varchar(50) |
| Email | Varchar(50) |
| RoleId (FK) | int |
| Password | Varchar(50) |

Role

|  |  |
| --- | --- |
| Id (PK) | Int |
| Title | Varchar(50) |

Holiday

|  |  |
| --- | --- |
| Id (PK) | Int |
| UserId | Varchar(100) |
| ContactInfo | Datae |
| DateOfHoliday | Date |
| CountGuests | Int |
| Place | Int |
| Priority | Int |
| SupervisorId (FK) | Int |
| Status | Bool |

Supervisor

|  |  |
| --- | --- |
| Id (PK) | Int |
| FirstName | Varchar(50) |
| SecondName | Varchar(50) |
| FatherName | Varchar(50) |
| ContactInfo | Varchar(100) |

HolidayHistory

|  |  |
| --- | --- |
| Id (PK) | Int |
| UserId (FK) | Int |
| ContactInfo | Varchar(100) |
| StartDate | Date |
| EndDate | Date |

Скрипты создания таблиц.

Schema::create('users', function (Blueprint $table) {

$table->id();

$table->string('first\_name');

$table->string('last\_name');

$table->string('father\_name');

$table->string('email')->unique();

$table->timestamp('email\_verified\_at')->nullable();

$table->string('phone\_number')->unique();

$table->unsignedBigInteger('role\_id')->default(0);

$table->string('password');

$table->rememberToken();

$table->timestamps();

$table->softDeletes();

$table->foreign('role\_id')->references('id')->on('roles');

});



Рисунок 5 – Таблица пользователей в БД

Schema::create('holidays', function (Blueprint $table) {  
 $table->id();  
 $table->unsignedBigInteger('user\_id');  
 $table->string('contact\_info');  
 $table->date('date\_of\_holiday');  
 $table->integer('count\_of\_guests');  
 $table->string('place');  
 $table->boolean('status');  
 $table->integer('priority');  
 $table->unsignedBigInteger('supervisor\_id');  
 $table->timestamps();  
 $table->softDeletes();  
  
 $table->foreign('user\_id')->references('id')->on('users');  
 $table->foreign('supervisor\_id')->references('id')->on('supervisors');  
});

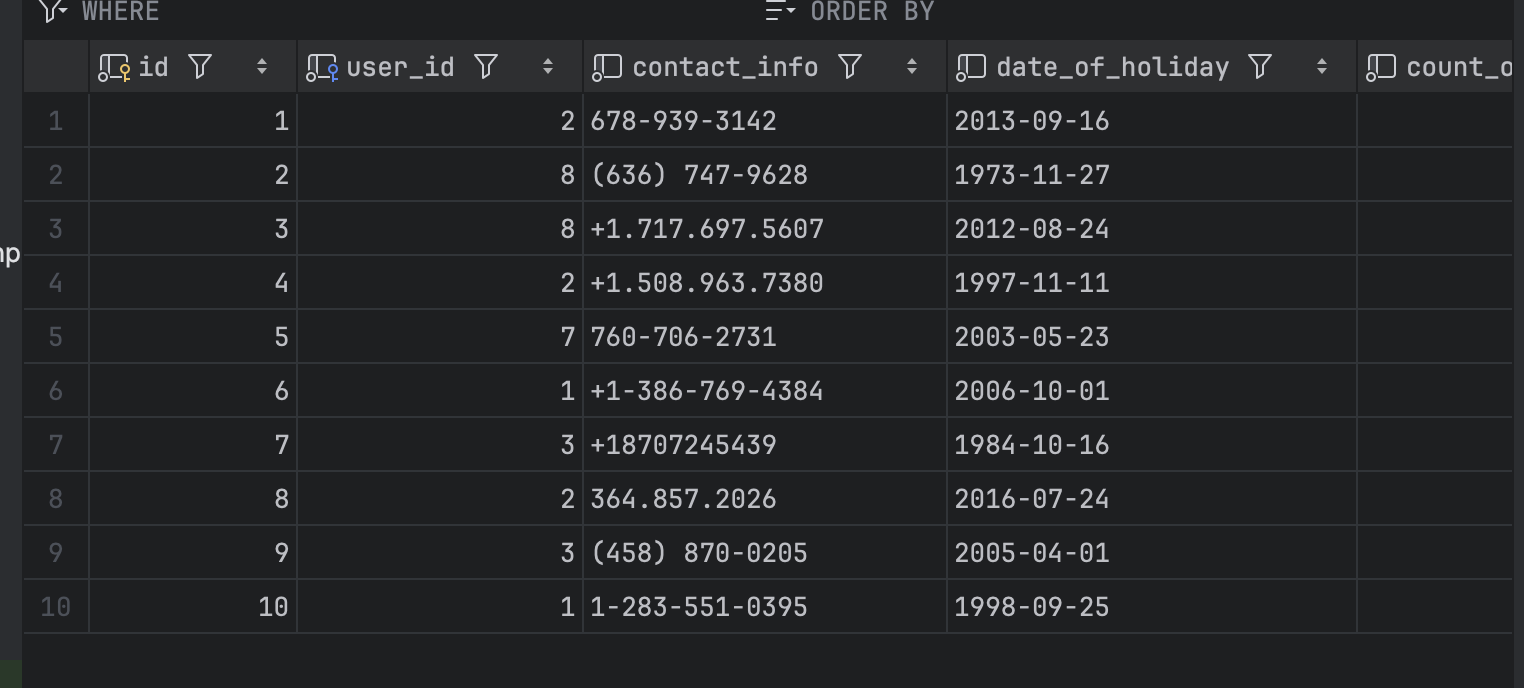


Рисунок 6 – Таблица праздников в БД

Schema::create('roles', function (Blueprint $table) {  
 $table->id();  
 $table->string('title');  
 $table->timestamps();  
 $table->softDeletes();  
});

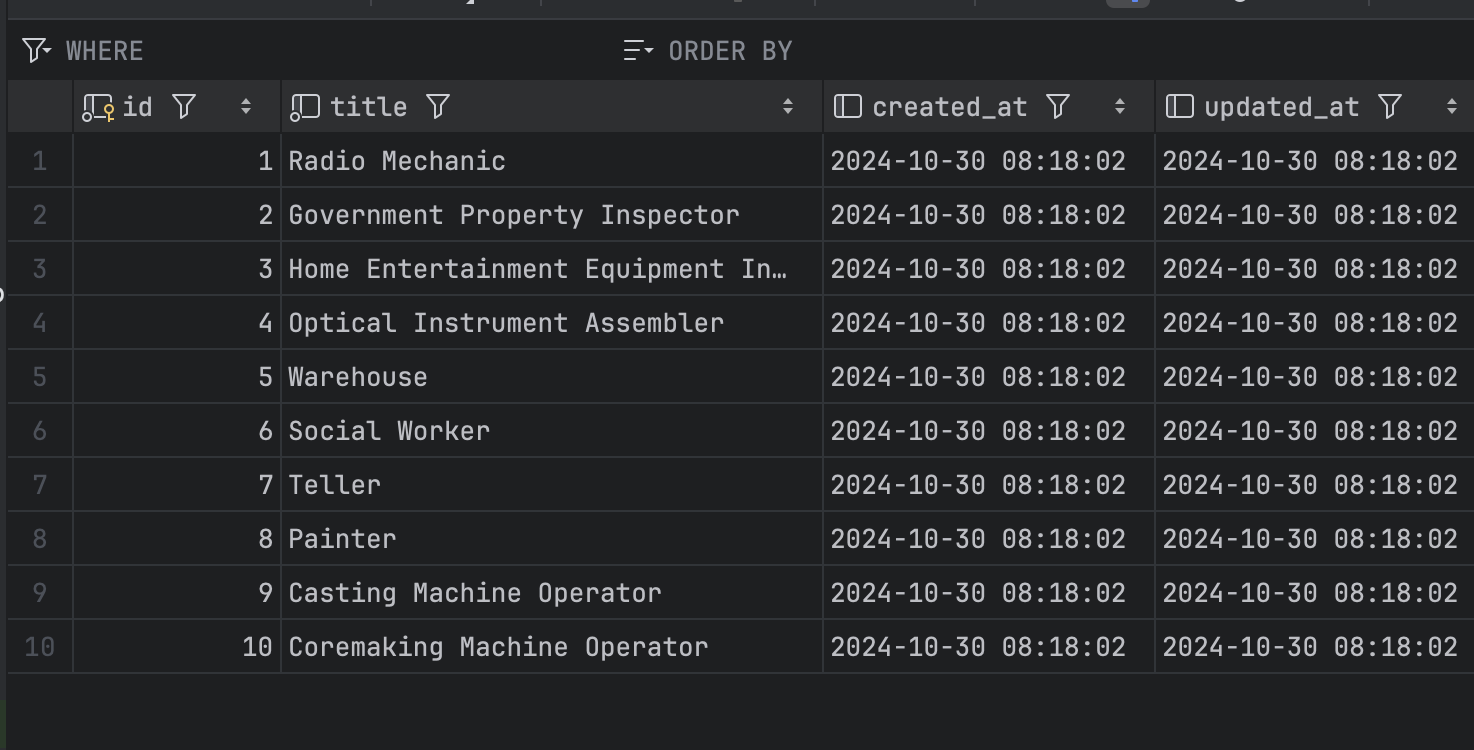


Рисунок 7 – Таблица ролей в БД

Schema::create('supervisors', function (Blueprint $table) {  
 $table->id();  
 $table->string('first\_name');  
 $table->string('second\_name');  
 $table->string('father\_name');  
 $table->string('contact\_info');  
 $table->softDeletes();  
});

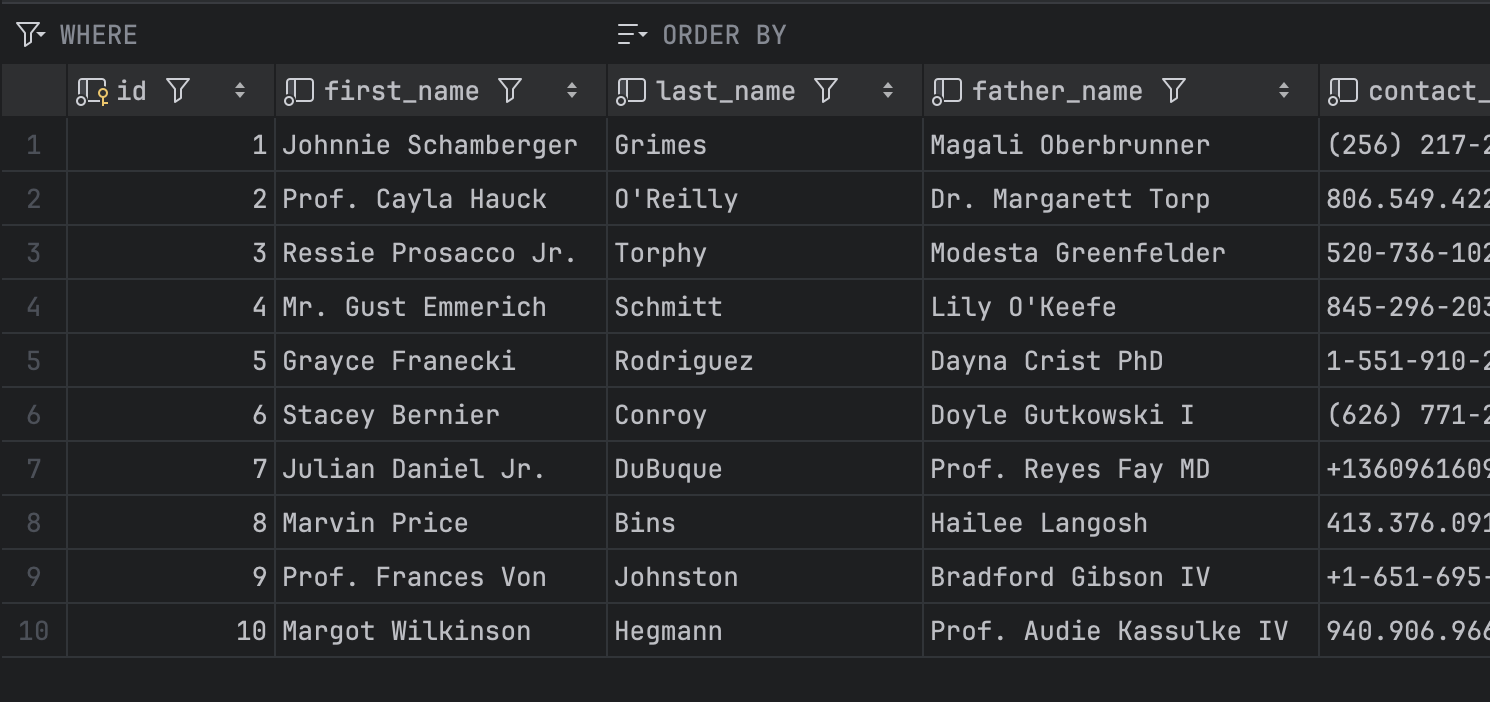


Рисунок 8 – Таблица организаторов в БД

Schema::create('holidays\_histories', function (Blueprint $table) {  
 $table->id();  
 $table->unsignedBigInteger('user\_id');  
 $table->string('contact\_info');  
 $table->date('start\_date');  
 $table->date('end\_date');  
 $table->timestamps();  
 $table->softDeletes();  
  
 $table->foreign('user\_id')->references('id')->on('users');  
});

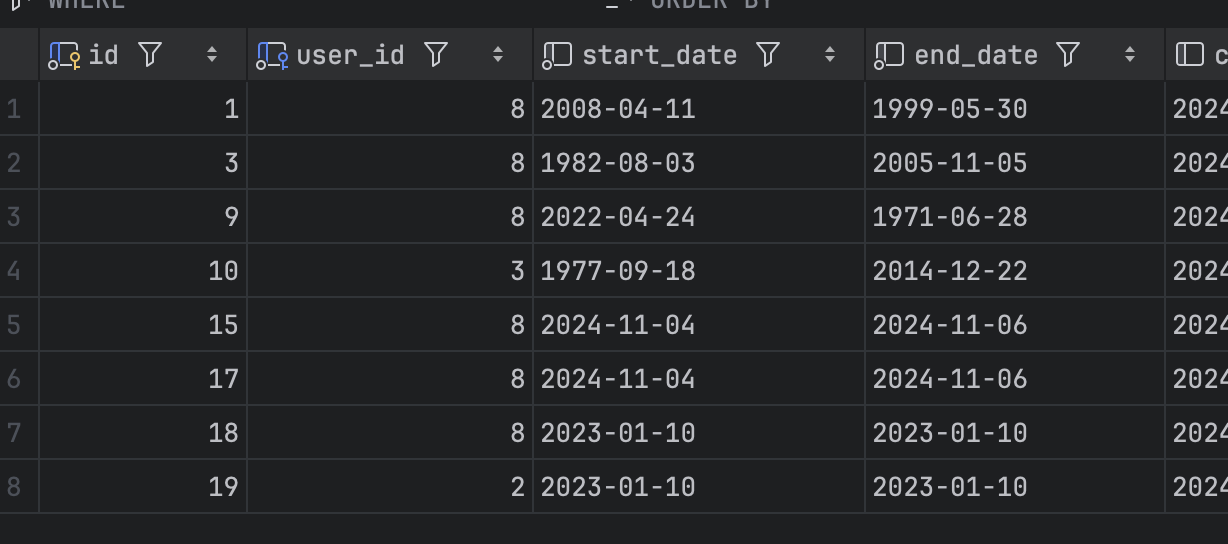


Рисунок 9 – Таблица истории заказов в БД

Скрипты связей.

User.

public function holidays()  
{  
 return $this->hasMany(Holiday::class);  
}  
public function role()  
{  
 return $this->belongsTo(Role::class);  
}  
  
public function holidaysHistory()  
{  
 return $this->hasMany(HolidayHistory::class);  
}

Holiday.

public function supervisor()  
{  
 return $this->belongsTo(Supervisor::class);  
}

Supervisor.

public function holidays()  
{  
 return $this->hasMany(Holiday::class);  
}

Role.

public function user()  
{  
 return $this->hasMany(User::class);  
}

HolidayHistory.

public function user()  
{  
 return $this->belongsTo(User::class);  
}

Проектирование интерфейса пользователя.

Для проектирования пользовательского интерфейса будем использовать варйрем – черно-белый подробный план страницы сайта. Здесь намечается расположение элементов: кнопок, изображений, текстов, и мокап – красивый вариант вайрфрейма. Тут уже появляются цвета, подбираются изображения, продумывается типографика. Получается красивая картинка приложения или сайта.

На рисунках 10 -12 представлены варфреймы страниц бущего веб-приложения.

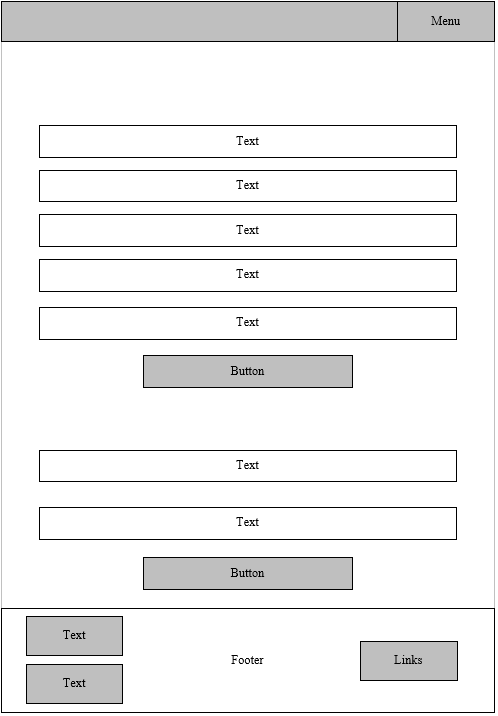


Рисунок 10 – Варфрейм профиля

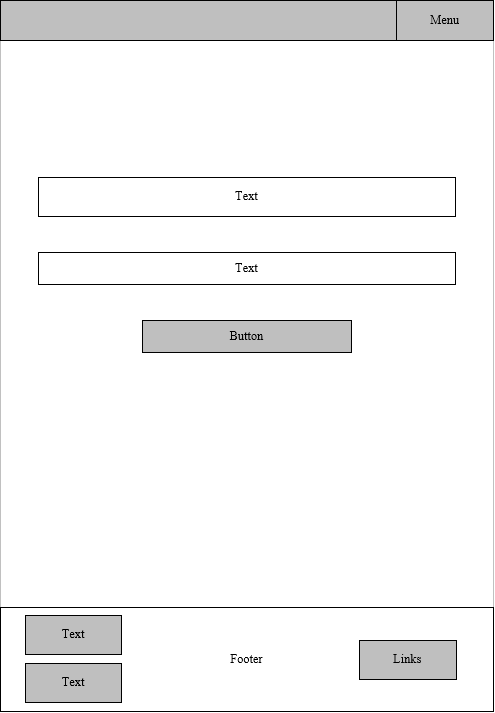


Рисунок 11 – Варфрейм входа

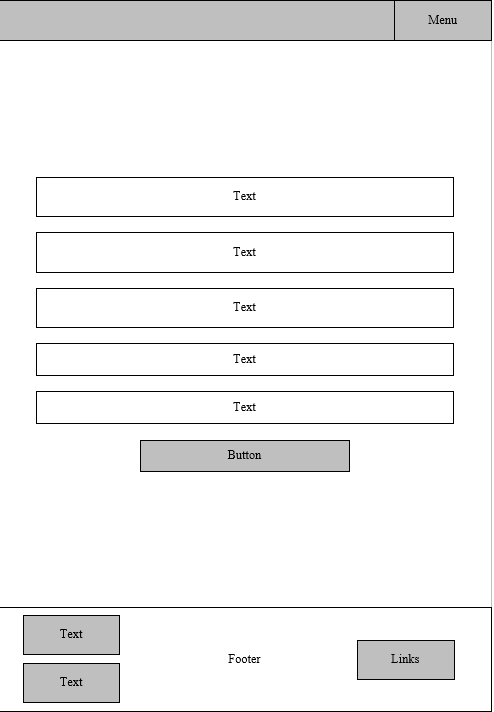


Рисунок 12 – Варфрейм регистрации

На рисунках 12 – 14 представлены мокапы страниц веб-приложения.

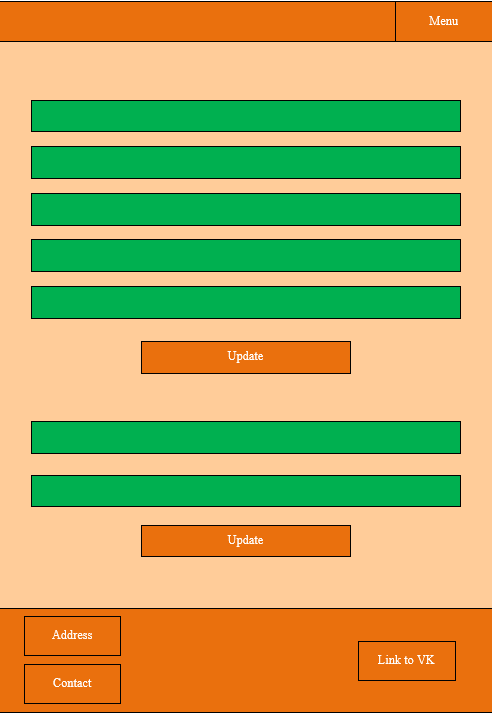


Рисунок 12 – Мокап профиля

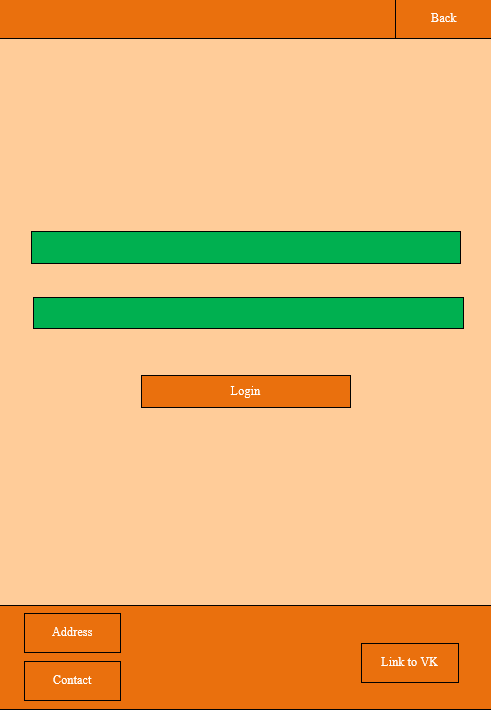


Рисунок 13 – Мокап входа

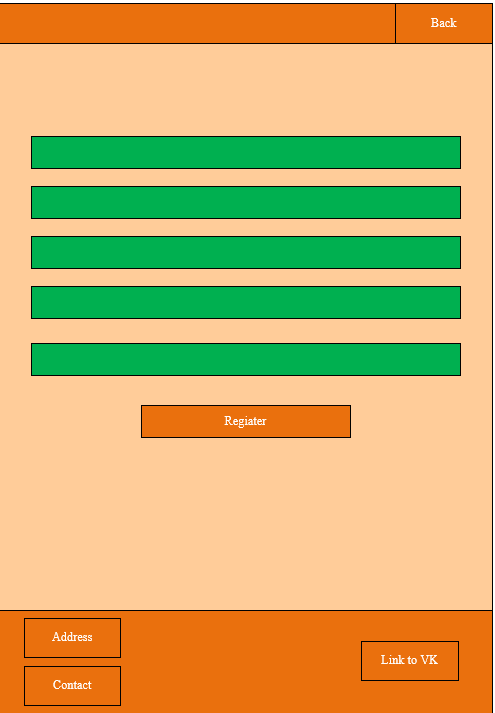


Рисунок 14 – Мокап регистрации

Так же не мало важной частью является карта навигации, которая определит основные маршруты в приложении (Рисунок 15).

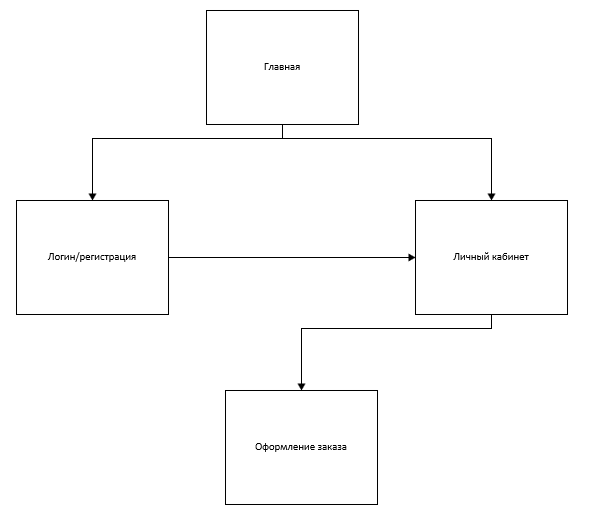


Рисунок 15 – Карта навигации

Разработка модели данных. Разработка программного продукта. Создание функциональности ПО.

Разработка веб приложения начинается с определения маршрутов, которые будут присутствовать в приложении. Все прописанные маршруты в Laraevl находятся в файле web.php. Самый просто маршрут содержит в себе непосредственно URI – строка, позволяющая идентифицировать какой-либо ресурс, и контроллер (можно заменить на callback функцию). Так же можно задать имя маршту через метод name(), на вход который принимает будущее имя маршрута. Конвенция названий маршрутов есть в официальной документации Laravel, но её можно не придерживаться и называть маршруты по-своему

Код, представленный ниже, описывает маршруты в приложении для праздничного агентства.

<?php  
  
use Illuminate\Support\Facades\Route;  
  
Route::get('/', function () {  
 return view('welcome');  
});  
  
Auth::*routes*();  
  
Route::get('/home', [App\Http\Controllers\HomeController::class, 'index'])->name('home');  
  
Route::group(['middleware'=>'checkProfileAccess','namespace' => 'App\Http\Controllers\Profile'], function () {  
 Route::get('/profile/{user}', 'IndexController')->name('profile.index');  
 Route::patch('/profile/{user}/update-profile', 'UpdateController')->name('profile.update');  
 Route::patch('/profile/{user}/update-password', 'UpdatePasswordController')->name('profile.update.password');  
});  
  
Route::group(['namespace' => 'App\Http\Controllers\Holiday'], function () {  
 Route::get('/home/create-holiday', 'CreateController')->name('holidays.create');  
 Route::post('/home/store-holiday', 'StoreController')->name('holidays.store');  
});  
  
Route::group(['middleware' => 'checkModeratorAccess','namespace' => 'App\Http\Controllers\Moderator'], function () {  
 Route::get('/moderator/users', 'IndexController')->name('moderator.index');  
 Route::get('/moderator/users/{user}/edit-holiday', 'EditController')->name('moderator.holiday.edit');  
 Route::patch('/moderator/users/{user}/update-holiday', 'UpdateController')->name('moderator.holiday.update');  
});  
  
Route::group(['middleware' => 'checkAdminAccess','namespace' => 'App\Http\Controllers\Admin'], function () {  
 Route::get('/admin/users', 'IndexController')->name('admin.index');  
 Route::get('/admin/users/{user}/edit', 'EditController')->name('admin.edit');  
 Route::patch('/admin/users/{user}/update', 'UpdateController')->name('admin.update');  
 Route::patch('/admin/users/{user}/update-password', 'UpdatePasswordController')->name('admin.update.password');  
 Route::delete('/admin/users/{user}/delete', 'DeleteController')->name('admin.destroy');  
 Route::get('/admin/users/create', 'CreateController')->name('admin.create');  
 Route::post('/admin/users/store', 'StoreController')->name('admin.store');  
});  
  
Route::fallback(function () {  
 return back();  
});

Чтобы реализовать логику приложения нам потребуется контроллер и модель. Контроллер отвечает за саму логику, модель отвечает за данные, методы работы с ними и структуру программы. Из скриптов выше у нас уже есть таблица пользователей, заполненная тестовыми данными, так как модель, при php artisan make:model -m, создаётся сразу с миграцией, нужной для создания БД. Далее пишем строчки кода нужные для отображения данным в профиле.

<?php  
namespace App\Http\Controllers\Profile;  
use App\Http\Controllers\Controller;  
use App\Models\User;  
use Illuminate\Http\Request;  
use Illuminate\Support\Facades\Auth;  
  
class IndexController extends Controller  
{  
 public function \_\_invoke(User $user)  
 {  
 return view('profile.index', compact('user'));  
 }  
}

Метод \_\_invoke, будет выполняться при вызове методе, делая объект функцией. Такой подход называется «Однометодные контроллеры», но это не обязатльеон и вместо Route::get('/profile/{user}', 'IndexController')->name('profile.index') можно было написать Route::get('/profile/{user}', [ProfileController::class, 'Index'])->name('profile.index'). В таком случае был бы один контроллер ProfileController, внутри которого были бы все остальные методы. Так как мы используем однометодные контроллеры, то струкрута папок выглядит как показано на рисунке 16.

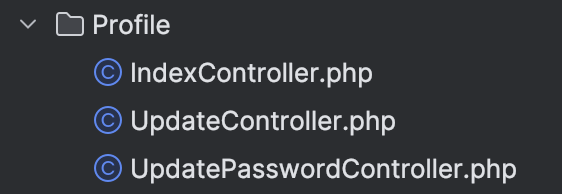


Рисунок 16 – Структура папок

Так как в котроллер возвращает страницу, то нам необходимо создать данную страницу. Создавать нужно в папке «Resources», затем «Views», здесь создать папку «profile» и в ней создать страницы, относящиеся к профилю.

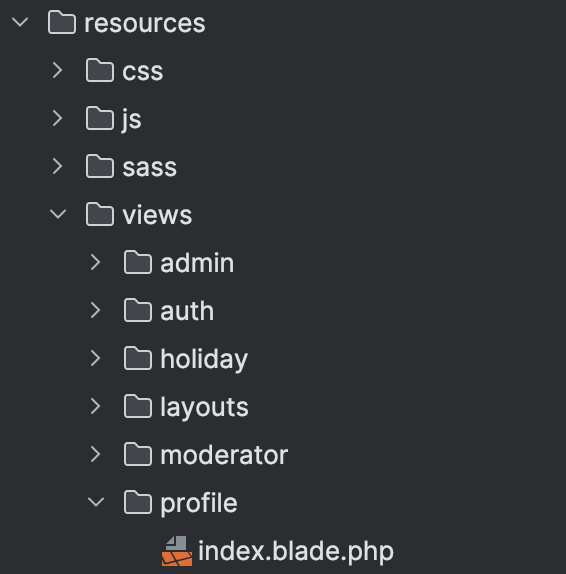


Рисунок 17 – Структура папок в ресурсах

Модульное тестирование. Создание тестовых случаев.

|  |  |
| --- | --- |
| **Название проекта** | Праздничное агентство |
| **Рабочая версия** | 1.0 |
| **Имя тестирующего** | Орлов Александр |
| **Дата(ы) теста** | 06.11.2024 |

|  |  |
| --- | --- |
| **Тестовый пример #** | 1 |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Подтвердить станицу регистрации |
| **Краткое изложение теста** | Осуществить успешную авторизацию |
| **Этапы теста** | 1. Заполнить поля соответствующими данными 2. Отправить форму нажатием на кнопку «Зарегистрироваться» |
| **Тестовые данные** | Имя: Александр  Фамилия: Александрович  Отчество: Александрович  Номер телефона: 8911111111  Адрес почты: [example@test.com](mailto:example@test.com)  Пароль: 123123  Подтверждение пароля: 123456 |
| **Ожидаемый результат** | Ошибки регистрации, из-за несходства паролей |
| **Фактический результат** | Ошибка регистрации |
| **Статус** | Зачет |
| **Предварительное условие** | Пользователь должен находиться на странице регистрации |
| **Постусловие** | После выполнения, программа должна работать в штатном режиме |
| **Примечания/комментарии** |  |

|  |  |
| --- | --- |
| **Тестовый пример #** | 2 |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Подтвердить станицу входа |
| **Краткое изложение теста** | Осуществить успешную авторизацию |
| **Этапы теста** | 1. Заполнить поля соответствующими данными 2. Отправить форму нажатием на кнопку «Войти» |
| **Тестовые данные** | Адрес почты: [gibson.geovanny@example.com](mailto:gibson.geovanny@example.com)  Пароль: password |
| **Ожидаемый результат** | Успешная авторизация |
| **Фактический результат** | Успешная авторизации |
| **Статус** | Зачет |
| **Предварительное условие** | Пользователь должен находиться на странице входа |
| **Постусловие** | После выполнения, программа должна работать в штатном режиме |
| **Примечания/комментарии** |  |

|  |  |
| --- | --- |
| **Тестовый пример #** | 3 |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Осуществление заказа меропритятия |
| **Краткое изложение теста** | Осуществить успешную авторизацию |
| **Этапы теста** | 1. Пройти регистрацию/авторизацию 2. Перейти на страницу заказа мероприятия 3. Заполнить поля соответствующими данными 4. Отправить форму нажатием на кнопку «Отправить заявку» |
| **Тестовые данные** | Контактная информация: 891111111  Дата: 22.02.2022  Число гостей: 100  Место проведения: 100 улица, г. Лос-Анджелес |
| **Ожидаемый результат** | Успешная отправка заявки на мероприятие |
| **Фактический результат** | Успешная отправка заявки на мероприятие |
| **Статус** | Зачет |
| **Предварительное условие** | Пользователь должен находиться в системе |
| **Постусловие** | После выполнения, программа должна работать в штатном режиме |
| **Примечания/комментарии** |  |

|  |  |
| --- | --- |
| **Тестовый пример #** | 4 |
| **Приоритет тестирования** | Средний |
| **Заголовок/название теста** | Удаление пользователя |
| **Краткое изложение теста** | Пользователь с ролью администратора удаляет пользователя |
| **Этапы теста** | 1. Пройти авторизацию 2. Перейти на страницу администратора 3. Выбрать нужного пользователя 4. Нажать на кнопку «Удалить» |
| **Тестовые данные** | - |
| **Ожидаемый результат** | Успешное удаление пользователя |
| **Фактический результат** | Успешное удаление пользователя |
| **Статус** | Зачет |
| **Предварительное условие** | Пользователь должен находиться в системе с правами администратора |
| **Постусловие** | После выполнения, программа должна работать в штатном режиме |
| **Примечания/комментарии** |  |

|  |  |
| --- | --- |
| **Тестовый пример #** | 5 |
| **Приоритет тестирования** | Средний |
| **Заголовок/название теста** | Окончание мероприятия пользователя |
| **Краткое изложение теста** | Пользователь с ролью модератора завершает праздни клиента |
| **Этапы теста** | 1. Пройти авторизацию 2. Перейти на страницу модератора 3. Выбрать нужного пользователя 4. Выбрать праздник, которые подошёл к концу 5. Нажать кнопку «Обновить» |
| **Тестовые данные** | - |
| **Ожидаемый результат** | Успешное завершение праздника пользователя |
| **Фактический результат** | Успешное завершение праздника пользователя |
| **Статус** | Зачет |
| **Предварительное условие** | Пользователь должен находиться в системе с правами модератора |
| **Постусловие** | После выполнения, программа должна работать в штатном режиме |
| **Примечания/комментарии** |  |

Создание тестовых случаев. Отладка программных модулей.

Для тестирования веб-приложения будем использовать команды – Command. Это php классы, которые имеют фукнцию handle, в которой происходит логика. Запускаются они непосредственно из командной строки с помощью php artisan.

Test 1.

try  
{  
 $userId = $this->ask('Введите Id пользователя');  
 $user = User::*findOrFail*($userId);  
 if ($user->role\_id >= 4)  
 print('Есть доступ к адмистрированию.');  
 else  
 print('Доступа к администрированию нет');  
}  
catch (\Exception $e)  
{  
 $this->error('Пользователя с Id '.$userId.' нет');  
}

При успешном прохождении теста (Рисунок 18), то есть при существующем пользователе с введённым Id и имеющем доступ к админ панели, распечатается «'Есть доступ к адмистрированию», иначе «'Доступа к администрированию нет», а если такого пользователя нет - «'Пользователя с Id '.$userId.' нет'».

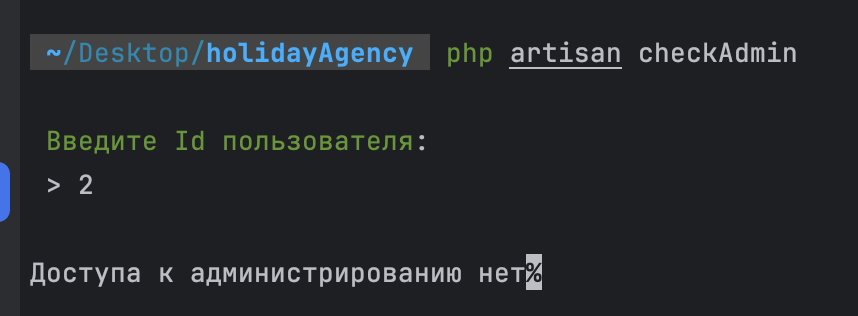


Рисунок 18 – Тестирование

Test 2.

try  
{  
 $userId = $this->ask('Введите Id пользователя');  
 $user = User::*findOrFail*($userId);  
 if ($user->role\_id >= 2)  
 print('Есть доступ к домерации.');  
 else  
 print('Доступа к модерации нет');  
}  
catch (\Exception $e)  
{  
 $this->error('Пользователя с Id '.$userId.' нет');  
}

Если пользователь имеет доступ к панели модератора, то тест пройден, иначе распечатается «Доступа к модерации нет» (Рисунок 19) или «Пользователя с Id '.$userId.' нет».

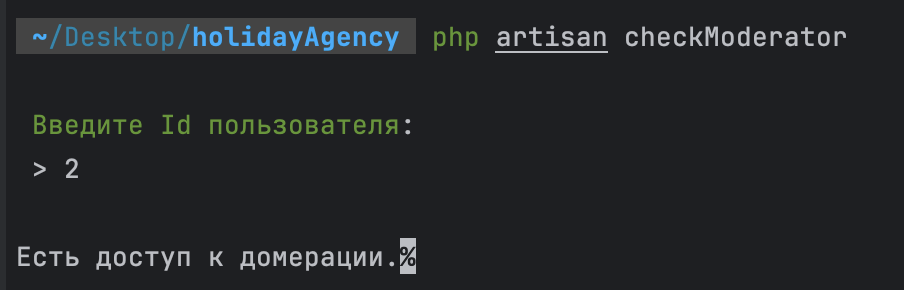


Рисунок 19 – Тестирование

Test 3.

try  
{  
 $data = [  
 'user\_id' => User::*all*()->random()->id,  
 'contact\_info' => '41234fsdfas',  
 'date\_of\_holiday' => '2023-01-10',  
 'count\_of\_guests' => 100,  
 'place' => 'dfgdfghgfdgfd',  
 'status' => 1,  
 'priority' => 1,  
 'supervisor\_id' => Supervisor::*all*()->random()->id,  
 ];  
 Holiday::*create*($data);  
 print('Успех!');  
}  
catch (\Exception $e)  
{  
 $this->error('Ошибка создания праздника!');  
}

При создании пользователя праздника в консоль печатается «Успех!» (Рисунок 20).

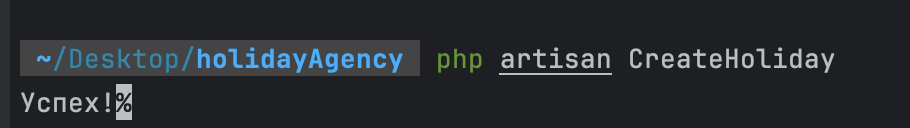


Рисунок 20 - Тестирование

Test 4.

try  
{  
 $data = [  
 'user\_id' => User::*all*()->random()->id,  
 'start\_date' => '2023-01-10',  
 'end\_date' => '2023-01-10',  
 ];  
 HolidayHistory::*create*($data);  
 print('Успех!');  
}  
catch (\Exception $e)  
{  
 $this->error('Ошибка создания истории праздника!');  
}

При успешном создании истории праздников в консоль печатается «Успех!» (Рисунок 21).

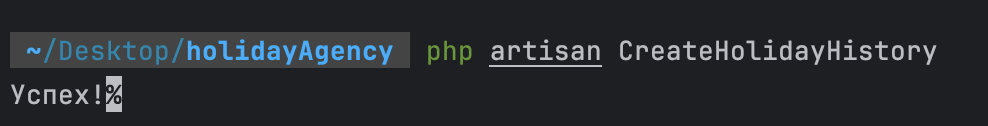


Рисунок 21 – Тестирование

Test 5.

try  
{  
 $data = [  
 'title' => 'super admin',  
 ];  
 Role::*create*($data);  
 print('Успех!');  
}  
catch (\Exception $e)  
{  
 $this->error('Ошибка создания роли!');  
}

При успешном создании рои в консоль печатается «Успех!» (Рисунок 22).

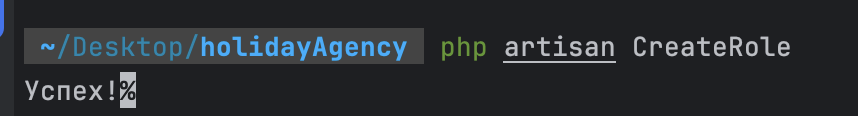


Рисунок 22 – Тестирование

Test 6.

try {  
 $data = [  
 'first\_name' => 'John Doe',  
 'last\_name' => 'Doe',  
 'father\_name' => 'John Doe',  
 'contact\_info' => 'john.doe@example.com',  
 ];  
 Supervisor::*create*($data);  
 print('Успех!');  
}  
catch (\Exception $e)  
{  
 $this->error('Ошибка создания супервайзера');  
}

При успешном создании ответственного за праздник в консоль печатается «Успех!» (Рисунок 23).

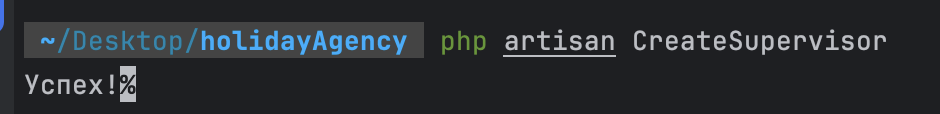


Рисунок 23 – Тестирование

Test 7.

try  
{  
 $data = [  
 'first\_name' => 'John Doe',  
 'last\_name' => 'Doe',  
 'father\_name' => 'John Doe',  
 'email' => 'john.doe@example.com',  
 'phone\_number' => '0123456789',  
 'password' => Hash::make('123123'),  
 ];  
 User::*create*($data);  
 print('Успех!');  
}  
catch (\Exception $e)  
{  
 $this->error('Неудалось создать пользователя');  
}

При успешном создании пользователя в консоль печатается «Успех!» (Рисунок 24).

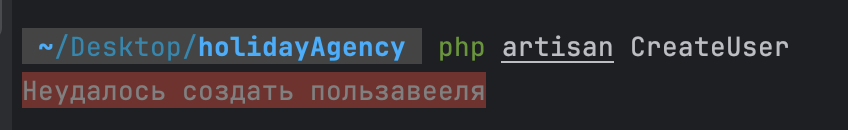


Рисунок 24 - Тестирование

Test 8.

try  
{  
 $userId = $this->ask('Введите id пользователя?');  
 $user = User::*findOrFail*($userId);  
 print\_r($user);  
}  
catch (\Exception $e)  
{  
 $this->error('Пользователя с i'.$userId.' нет');  
}

При успешном поиске пользователя, он распечатается в консоль (Рисунок 25).

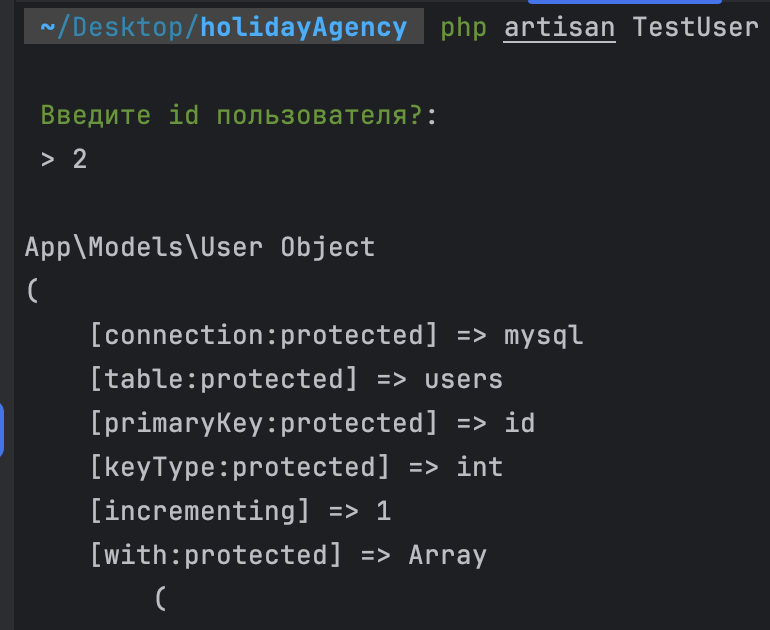


Рисунок 25 – Тестирование

Test 9.

try  
{  
 $userId = $this->ask('Введите Id пользователя');  
 $user = User::*findOrFail*($userId);  
 if ($user->holidays->count() > 0)  
 print('Есть заказанные праздники');  
 else  
 print('Нет заказанных праздников');  
}  
catch (\Exception $e)  
{  
 $this->error('Пользователя с Id '.$userId.' нет');  
}

При успешном поиске праздников пользователя в консоль печатается «Есть заказанные праздники» (Рисунок 26).

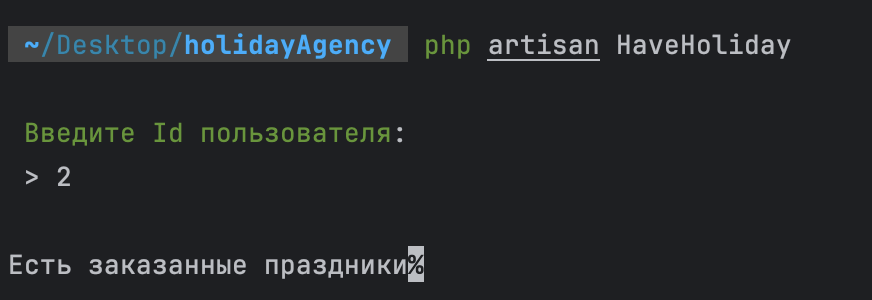


Рисунок 26 - Тестирование

Test 10.

try  
{  
 $userId = $this->ask('Введите Id пользователя');  
 $user = User::*findOrFail*($userId);  
 if ($user->holidaysHistory->count() > 0)  
 print('Были ранее аказы');  
 else  
 print('Нет истории заказов');  
}  
catch (\Exception $e)  
{  
 $this->error('Пользователя с Id '.$userId.' нет');  
}

При успешном поиске истории праздников пользователя в консоль печатается «Есть заказанные праздники» (Рисунок 27).

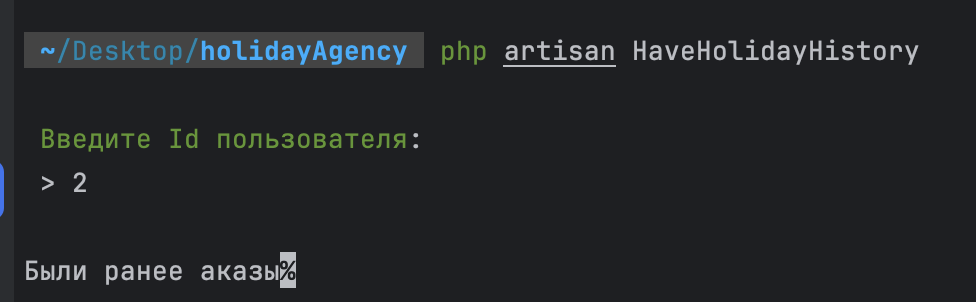


Рисунок 27 – Тестирование

ЗАКЛЮЧЕНИЕ

В ходе учебной практики были освоены основные аспекты разработки программного продукта начиная от разработки технического задания и описания предметной области до тестоирования конечного продукта, написания тестовых случаев и разработки Unit тестов.

**АТТЕСТАЦИОННЫЙ ЛИСТ**

**ПО УЧЕБНОЙ ПРАКТИКЕ УП.02.01 (ПО ПРОФИЛЮ СПЕЦИАЛЬНОСТИ)**

по профессиональному модулю ПМ.02 «Осуществление интеграции программных модулей»

(код и наименование)

Специальность09.02.07 Информационные системы и программирование

(код и наименование специальности)

Студент(ка) 4 курса 42919/7 группы

Орлов Александр Александрович

(Фамилия, имя, отчество)

Место прохождения практики: УВЦ, пр. Энгельса, 23

(наименование и адрес организации)

Период прохождения практики

с «28» октября 2024 г. по «09» ноября 2024 г.

**Виды и качество выполнения работ**

|  |  |  |
| --- | --- | --- |
| **Виды выполненных работ обучающимся**  **во время практики** | **Объем работ, час.** | **Качество выполнения работ (оценка по пятибалльной системе)** |
| **Тема 1.** Участие в выработке требований к программному обеспечению. | 12 |  |
| **Тема 2.** Стадии проектирования программного обеспечения. Проектирование UML-диаграмм. | 15 |  |
| **Тема 3.** Конструирование пользовательского интерфейса. Разработка модулей программного обеспечения. | 25 |  |
| **Тема 4.** Тестирование и отладка программных модулей. Модульное тестирование. | 12 |  |
| **Тема 5.** Разработка программной документации. | 9 |  |

**Характеристика учебной/профессиональной деятельности обучающегося во время учебной практики (по профилю специальности):**

Общие и профессиональные компетенции, предусмотренные программой практики, освоены **/** не освоены.

(нужное подчеркнуть)

Итоговая оценка по практике \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Руководитель практики Хисамутдинова А.С. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Ф.И.О.) (подпись)

Дата «09» ноября 2024 г.