2.4 Разработка структуры информационной системы

Информационная система разработана с использованием технологии MVC. Каждый модуль системы состоит из классов-моделей, которые представляют собой имитацию соответствующих таблиц в базе данных, классов-контроллеров, которые содержат методы для работы с моделями и отображения данных пользователю, и веб-страниц, предоставляющих пользователю графический интерфейс.

В информационной системе имеются следующие классы.

Классы-модели:

- ActivityLevel – представляет таблицу activity_levels. Содержит методы для добавления, обновления, удаления и поиска уровней физической активности, а также для получения записей из связанных таблиц в виде объектов;
- Article – представляет таблицу articles. Содержит методы для добавления, обновления, удаления и поиска статей, а также для получения записей из связанных таблиц в виде объектов;
- ArticlesRating – представляет таблицу articles_ratings. Содержит методы для добавления, обновления, удаления и поиска оценок на статьи, а также для получения записей из связанных таблиц в виде объектов;
- DiaryEntry – представляет таблицу diary_entries. Содержит методы для добавления, обновления, удаления и поиска записей в дневниках пользователей, а также для получения записей из связанных таблиц в виде объектов;
- DietType – представляет таблицу diet_types. Содержит методы для добавления, обновления, удаления и поиска типов диет, а также для получения записей из связанных таблиц в виде объектов;

- Exercise – представляет таблицу exercises. Содержит методы для добавления, обновления, удаления и поиска упражнений, а также для получения записей из связанных таблиц в виде объектов;
- ExerciseCategory – представляет таблицу exercise_categories. Содержит методы для добавления, обновления, удаления и поиска категорий упражнений, а также для получения записей из связанных таблиц в виде объектов;
- ExerciseMeasurement – представляет таблицу exercise_measurements. Содержит методы для добавления, обновления, удаления и поиска единиц измерения упражнений, а также для получения записей из связанных таблиц в виде объектов;
- FavouriteExercise – представляет таблицу favourite_exercises. Содержит методы для добавления, обновления, удаления и поиска избранных упражнений пользователей, а также для получения записей из связанных таблиц в виде объектов;
- FavouriteProduct – представляет таблицу favourite_products. Содержит методы для добавления, обновления, удаления и поиска избранных продуктов пользователей, а также для получения записей из связанных таблиц в виде объектов;
- FavouriteRecipe – представляет таблицу favourite_recipes. Содержит методы для добавления, обновления, удаления и поиска избранных рецептов пользователей, а также для получения записей из связанных таблиц в виде объектов;
- FavouriteWorkout – представляет таблицу favourite_workouts. Содержит методы для добавления, обновления, удаления и поиска избранных программ тренировок пользователей, а также для получения записей из связанных таблиц в виде объектов;
- FoodItem – представляет таблицу food_items. Содержит методы для добавления, обновления, удаления и поиска единиц пищи, а

также для получения записей из связанных таблиц в виде объектов;

- Goal – представляет таблицу goals. Содержит методы для добавления, обновления, удаления и поиска целей пользователей, а также для получения записей из связанных таблиц в виде объектов;
- PregnancyAndBreastfeedingType – представляет таблицу pregnancy_and_breastfeeding_types. Содержит методы для добавления, обновления, удаления и поиска типов беременности и грудного кормления, а также для получения записей из связанных таблиц в виде объектов;
- Product – представляет таблицу products. Содержит методы для добавления, обновления, удаления и поиска продуктов, а также для получения записей из связанных таблиц в виде объектов;
- ProductCategory – представляет таблицу product_categories. Содержит методы для добавления, обновления, удаления и поиска категорий продуктов, а также для получения записей из связанных таблиц в виде объектов;
- Recipe – представляет таблицу recipes. Содержит методы для добавления, обновления, удаления и поиска рецептов, а также для получения записей из связанных таблиц в виде объектов;
- RecipeCategory – представляет таблицу recipe_categories. Содержит методы для добавления, обновления, удаления и поиска категорий рецептов, а также для получения записей из связанных таблиц в виде объектов;
- RecipesProduct – представляет таблицу recipes_products. Содержит методы для добавления, обновления, удаления и поиска продуктов в рецептах, а также для получения записей из связанных таблиц в виде объектов;

- RecipesStage – представляет таблицу recipes_stages. Содержит методы для добавления, обновления, удаления и поиска этапов приготовления рецептов, а также для получения записей из связанных таблиц в виде объектов;
- Report – представляет таблицу reports. Содержит методы для добавления, обновления, удаления и поиска жалоб, а также для получения записей из связанных таблиц в виде объектов;
- Role – представляет таблицу roles. Содержит методы для добавления, обновления, удаления и поиска ролей, а также для получения записей из связанных таблиц в виде объектов;
- SportItem – представляет таблицу sport_items. Содержит методы для добавления, обновления, удаления и поиска единиц физической активности, а также для получения записей из связанных таблиц в виде объектов;
- User – представляет таблицу users. Содержит методы для добавления, обновления, удаления и поиска пользователей, а также для получения записей из связанных таблиц в виде объектов;
- Workout – представляет таблицу workouts. Содержит методы для добавления, обновления, удаления и поиска программ тренировок, а также для получения записей из связанных таблиц в виде объектов;
- WorkoutCategory – представляет таблицу workout_categories. Содержит методы для добавления, обновления, удаления и поиска категорий программ тренировок, а также для получения записей из связанных таблиц в виде объектов;
- WorkoutsExercise – представляет таблицу workouts_exercises. Содержит методы для добавления, обновления, удаления и поиска упражнений в программах тренировок, а также для получения записей из связанных таблиц в виде объектов.

Классы-контроллеры:

- ArticleController – содержит методы для добавления, редактирования, оценки и удаления статей;
- DiaryController – содержит методы для добавления записей в дневник, редактирования и удаления записи за текущий день;
- ExerciseController – содержит методы для добавления, редактирования и удаления упражнений, добавления упражнений в избранное и удаления их из избранного;
- PageController – содержит методы для поиска и выборки данных из БД и их передачи в шаблоны веб-страниц для отображения пользователю;
- ProductController – содержит методы для добавления, редактирования и удаления продуктов, добавления продуктов в избранное и удаления их из избранного;
- RecipeController – содержит методы для добавления, редактирования и удаления рецептов, добавления рецептов в избранное и удаления их из избранного;
- ReportController – содержит методы для отправки жалоб и их решения;
- UserController – содержит методы для регистрации пользователей, авторизации и выхода, редактирования данных профиля, расчёта текущей калорийной нормы пользователя;
- WorkoutController – содержит методы для добавления, редактирования и удаления программ тренировок, добавления программ тренировок в избранное и удаления их из избранного.

Рассмотрим подробнее логику работы некоторых классов.

Модуль учёта пользователей содержит подсистему регистрации, которая включает в себя страницу регистрации с формой для ввода данных и метод register класса UserController. Логика работы подсистемы представлена на рисунке 2.7.

```php
public function register(Request $request) {
    $userData = $request->all();

    $validator = Validator::make($userData, [
        'email' => 'required|unique:users|email:rfc,dns,filter',
        'password' => 'required',
        'password-repeat' => 'required|same:password',
        'name' => 'required',
        'birth-date' => 'required|date',
        'userpic' => 'mimes:jpeg,png,gif',
        'height' => 'required|numeric',
        'weight' => 'required|numeric',
    ], [
        'required' => 'Это поле не может быть пустым.',
        'unique' => 'Пользователь с таким email-адресом уже существует.',
        'email' => 'Введите корректный email.',
        'same' => 'Пароли не совпадают.',
        'date' => 'Введите корректную дату.',
        'mimes' => 'Фото должно быть в формате JPEG, PNG или GIF.',
        'numeric' => 'Введите число.',
    ]);

    if ($validator->fails()) {
        return back()
            ->withErrors($validator)
            ->withInput();
    }

    $user = new User();
    $user->name = $userData['name'];
    $user->gender = $userData['gender'];
    $user->birth_date = $userData['birth-date'];
    $user->height = $userData['height'];
    $user->weight = $userData['weight'];
    if ($userData['gender'] == 2) {
        $user->pregnancy_and_breastfeeding_type_id = $userData['pregnancy-and-breastfeeding-type'];
    } else {
        $user->pregnancy_and_breastfeeding_type_id = 1;
    }
    $user->activity_level_id = $userData['activity-level'];
    if (array_key_exists( key: 'activity-level-accounting', $userData)) {
        $user->activity_level_accounting = 1;
    } else {
        $user->activity_level_accounting = 0;
    }
    $user->goal_id = $userData['goal'];
    $user->diet_type_id = $userData['diet-type'];
    if (array_key_exists( key: 'public-diary', $userData)) {
        $user->public_diary = 1;
    } else {
        $user->public_diary = 0;
    }
    if (!is_null($userData['personal-info'])) {
        $user->personal_info = $userData['personal-info'];
    }
    if (array_key_exists( key: 'userpic', $userData)) {
        $user->userpic_url = $userData['userpic']->store('img/uploads/userpics');
    }
    $user->email = $userData['email'];
    $user->password = bcrypt($userData['password']);
    $user->save();

    return redirect(route( name: 'index-page'));
}
```

Рисунок 2.7 – Метод register

Метод принимает данные, введённые пользователем, из объекта $request класса Request (это класс фреймворка Laravel, предназначенный для упрощения работы с вводом данных). Затем производится их валидация. Если валидация не прошла, то происходит перенаправление обратно на страницу

регистрации с сообщениями об ошибках. В ином случае происходит добавление записи с соответствующими данными в таблицу пользователей и перенаправление на главную страницу.

Модуль учёта дневников содержит метод createEntry для добавления новой записи в дневник. Часть данного метода, находящаяся после проведения валидации, отражена на рисунке 2.8.

```
95          $diaryEntry = new DiaryEntry();
96          $diaryEntry->author_id = Auth::user()->id;
97          $diaryEntry->target_calorie_amount = $this->calculateUserTargetCalorieAmount( goalAccounting: true);
98          $diaryEntry->normal_calorie_amount = $this->calculateUserTargetCalorieAmount( goalAccounting: false);
99          $diaryEntry->user_goal_id = Auth::user()->goal_id;
100         $diaryEntry->user_weight = Auth::user()->weight;
101         if (!is_null($diaryEntryData['comment'])) {
102             $diaryEntry->comment = $diaryEntryData['comment'];
103         }
104         $diaryEntry->save();
105         $id = $diaryEntry->id;
106
107         // Создаём FoodItems
108         for ($i = 0; $i < count($foodNames); $i++) {
109             $index = $i + 1;
110
111             $foodItem = new FoodItem();
112             $foodItem->entry_id = $id;
113             if (Product::where('name', array_values($foodNames)[$i])->get()->isNotEmpty()) {
114                 $productId = Product::where('name', array_values($foodNames)[$i])->first()->id;
115                 $foodItem->product_id = $productId;
116             } else {
117                 $recipeId = Recipe::where('name', array_values($foodNames)[$i])->first()->id;
118                 $foodItem->recipe_id = $recipeId;
119             }
120             $foodItem->index_number = $index;
121             $foodItem->food_weight = array_values($foodWeights)[$i];
122             $foodItem->save();
123         }
124
125         // Создаём SportItems
126         for ($i = 0; $i < count($sportsNames); $i++) {
127             $index = $i + 1;
128
129             $sportItem = new SportItem();
130             $sportItem->entry_id = $id;
131             if (Exercise::where('name', array_values($sportsNames)[$i])->get()->isNotEmpty()) {
132                 $exerciseId = Exercise::where('name', array_values($sportsNames)[$i])->first()->id;
133                 $sportItem->exercise_id = $exerciseId;
134             } else {
135                 $workoutId = Workout::where('name', array_values($sportsNames)[$i])->first()->id;
136                 $sportItem->workout_id = $workoutId;
137             }
138             $sportItem->index_number = $index;
139             $sportItem->sport_quantity = array_values($sportsQuantities)[$i];
140             $sportItem->save();
141         }
142
143         return redirect(route( name: 'diary-page'));
144     }
```

Рисунок 2.8 – Часть метода createEntry

В методе создаётся новая запись дневника, заполняется полученными данными и записывается в БД. Далее для всех добавленных в запись единиц пищи и физической активности создаются соответствующие записи в БД и заполняются принятыми данными. Затем происходит перенаправление на страницу дневника пользователя.

Модули учёта продуктов, рецептов, упражнений и программ тренировок имеют достаточно похожую логику, включающую в себя добавление новой записи, редактирование и удаление записи по её идентификатору, добавление записи в избранное и удаление из избранного. Рассмотрим это всё на примере класса ProductController, отвечающего за работу с продуктами.

Метод для добавления продукта называется createProduct и представлен на рисунке 2.9.

```php
public function createProduct(Request $request) {
    $productData = $request->all();

    $validator = Validator::make($productData, [
        'name' => 'required|unique:products',
        'photo' => 'mimes:jpeg,png,gif',
        'calories' => 'required|numeric',
        'proteins' => 'required|numeric',
        'fats' => 'required|numeric',
        'carbohydrates' => 'required|numeric',
    ], [
        'required' => 'Это поле не может быть пустым.',
        'unique' => 'Продукт с таким названием уже существует.',
        'mimes' => 'Фото должно быть в формате JPEG, PNG или GIF.',
        'numeric' => 'Введите число',
    ]);

    if ($validator->fails()) {
        return back()
            ->withErrors($validator)
            ->withInput();
    }

    $product = new Product();
    $product->author_id = Auth::user()->id;
    $product->category_id = $productData['category'];
    $product->name = $productData['name'];
    if (!is_null($productData['description'])) {
        $product->description = $productData['description'];
    }
    if (array_key_exists( key: 'photo', $productData)) {
        $product->photo_url = $productData['photo']->store('img/uploads/products');
    }
    $product->calories = $productData['calories'];
    $product->proteins = $productData['proteins'];
    $product->fats = $productData['fats'];
    $product->carbohydrates = $productData['carbohydrates'];
    if (array_key_exists( key: 'public', $productData)) {
        $product->public = 1;
    } else {
        $product->public = 0;
    }
    $product->save();

    $id = $product->id;

    return redirect(route( name: 'single-product-page', ['id' => $id]));
}
```

Рисунок 2.9 – Метод createProduct

Метод принимает данные от пользователя из объекта $request и производит их валидацию. В случае неуспешной валидации, пользователь попадает обратно на страницу добавления продукта, в которой отображаются

все допущенные ошибки. Иначе в таблицу продуктов добавляется новая запись с введёнными данными, после чего происходит перенаправление на страницу только что созданного продукта.

Метод для редактирования продукта называется editProduct и представлен на рисунке 2.10.

```
63      public function editProduct(Request $request, $id) {
64          $productData = $request->all();
65
66          $validator = Validator::make($productData, [
67              'name' => ['required', new UniqueInProductsExceptThis($id)],
68              'photo' => 'mimes:jpeg,png,gif',
69              'calories' => 'required|numeric',
70              'proteins' => 'required|numeric',
71              'fats' => 'required|numeric',
72              'carbohydrates' => 'required|numeric',
73          ], [
74              'required' => 'Это поле не может быть пустым.',
75              'mimes' => 'Фото должно быть в формате JPEG, PNG или GIF.',
76              'numeric' => 'Введите число',
77          ]);
78
79          if ($validator->fails()) {
80              return back()
81                  ->withErrors($validator)
82                  ->withInput();
83          }
84
85          $product = Product::findOrFail($id);
86          $product->category_id = $productData['category'];
87          $product->name = $productData['name'];
88          if (!is_null($productData['description'])) {
89              $product->description = $productData['description'];
90          } else {
91              $product->description = null;
92          }
93          if (array_key_exists( key: 'photo', $productData)) {
94              $product->photo_url = $productData['photo']->store('img/uploads/products');
95          }
96          $product->calories = $productData['calories'];
97          $product->proteins = $productData['proteins'];
98          $product->fats = $productData['fats'];
99          $product->carbohydrates = $productData['carbohydrates'];
100         if (array_key_exists( key: 'public', $productData)) {
101             $product->public = 1;
102         } else {
103             $product->public = 0;
104         }
105         $product->save();
106
107         return redirect(route( name: 'single-product-page', ['id' => $id]));
108     }
```

Рисунок 2.10 – Метод editProduct

Логика данного метода практически аналогична логике метода для создания продуктов. Из объекта $request принимаются данные от пользователя, и проводится их валидация. В случае неуспешной валидации

пользователь попадает обратно на страницу редактирования продукта. Иначе метод находит в базе данных нужный продукт по его идентификатору (который принимается в метод вместе с объектом $request), записывает в него новые данные и сохраняет в таблицу продуктов. После этого происходит перенаправление на страницу только что отредактированного продукта.

Метод для удаления продуктов называется removeProduct и отображён на рисунке 2.11.

```
125     public function removeProduct($id) {
126         $product = Product::findOrFail($id);
127         $product->deleted = 1;
128         $product->save();
129 
130         FavouriteProduct::where('product_id', $id)->delete();
131 
132         return redirect(route( name: 'products-page'));
133     }
```

Рисунок 2.11 – Метод removeProduct

Метод принимает идентификатор продукта, который он использует, чтобы найти нужный продукт в базе. Затем происходит пометка, что найденный продукт следует считать удалённым из системы, и запись сохраняется в базу данных. Далее этот продукт удаляется из избранного у всех пользователей посредством модели FavouriteProduct. В конце происходит переход на страницу со списком продуктов.

Ниже, на рисунке 2.12 представлены методы для добавления продукта в избранное и его удаления из избранного.

```
110     public function addFavouriteProduct($id) {
111         $favourite = new FavouriteProduct();
112         $favourite->user_id = Auth::user()->id;
113         $favourite->product_id = $id;
114         $favourite->save();
115 
116         return back();
117     }
118 
119     public function removeFavouriteProduct($id) {
120         FavouriteProduct::where('user_id', Auth::user()->id)->where('product_id', $id)->delete();
121 
122         return back();
123     }
```

Рисунок 2.12 – Методы addFavouriteProduct и removeFavouriteProduct

Методы принимают идентификатор продукта, который затем используют для реализации логики.

Для добавления продукта в избранное, создаётся новая запись в таблице избранных продуктов. В неё записывается идентификатор текущего авторизованного пользователя и идентификатор продукта, после чего происходит запись в БД и переход обратно на страницу продукта.

Для удаления продукта из избранного происходит поиск записи в таблице избранных продуктов с идентификатором текущего авторизованного пользователя и нужного продукта. Затем найденная запись удаляется из таблицы, и происходит переход на страницу продукта.

Рассмотрим один из методов класса PageController – метод singleWorkout для формирования массива данных о конкретной программе тренировок и передачи его на страницу программы тренировок. Данный метод представлен ниже на рисунке 2.13.

```
913     public function singleWorkout($id) {
914         $workout = Workout::findOrFail($id);
915 
916         if ($workout->deleted == 1) {
917             return view( view: 'single-workout-removed', ['workout' => $workout]);
918         }
919 
920         $workout->category_name = $workout->category()->first()->name;
921         $workout->description_paragraphs = collect(explode( delimiter: PHP_EOL, $workout->description));
922         if (Auth::check()) {
923             $workout->is_favourite = FavouriteWorkout::where('user_id', Auth::user()->id)
924                                                      ->where('workout_id', $workout->id)
925                                                      ->get()
926                                                      ->isNotEmpty();
927         }
928         $workout->author = User::findOrFail($workout->author_id);
929 
930         $workout->calorie_consumption = 0;
931 
932         $workout->exercises = collect([]);
933         $workoutExercises = WorkoutsExercise::where('workout_id', $workout->id)->get();
934         foreach ($workoutExercises as $workoutExercise) {
935             $exercise = Exercise::findOrFail($workoutExercise->exercise_id);
936             $exercise->quantity = $workoutExercise->exercise_quantity;
937             $exercise->measurement_name = $exercise->measurement()->first()->name;
938             $workout->exercises->push($exercise);
939 
940             $increment = $exercise->calorie_consumption * $exercise->quantity;
941             if ($exercise->user_weight_accounting == 1) {
942                 if (Auth::check()) {
943                     $increment *= Auth::user()->weight;
944                 } else {
945                     $increment *= 70; // average human weight
946                 }
947             }
948 
949             $workout->calorie_consumption += $increment;
950         }
951 
952         return view( view: 'single-workout', ['workout' => $workout]);
953     }
```

Рисунок 2.13 – Метод singleWorkout

Метод принимает в качестве параметра идентификатор, с помощью которого он получает из базы данных необходимую программу тренировок. Далее проверяется, не помечена ли данная программа тренировок как удалённая, и если да, то пользователь перенаправляется на страницу удалённой программы тренировок. Далее формируются данные, которые необходимо отображать на странице. Забирается название категории данной программы тренировок. Описание разбивается на абзацы для правильного отображения в HTML-разметке. Определяется, входит ли данная программа тренировок в избранное текущего авторизованного пользователя. Затем осуществляется формирование списка всех упражнений, входящих в данную программу тренировок, с попутным подсчётом её суммарных энергозатрат. После этого происходит переход на страницу программы тренировок, куда передаётся сформированный массив данных.