МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение высшего образования
«ИРКУТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(ФГБОУ ВО «ИГУ»)

Институт математики и информационных технологий

Кафедра информационных и алгебраических систем

ОТЧЕТ

о курсовой работе по курсу «Разработка WEB-приложений»
Разработка CRM-системы

Студента 3 курса группы 2371
Белогуба Константина Евгеньевича
Направление : 02.03.02 – Фундаментальная информатика и информационные технологии

Руководитель:
канд. техн. наук доцент
Черкашин Евгений Александрович

Курсовая работа защищена с оценкой

_____

Иркутск – 2021

Оглавление

Введение

При найме разработчиков приходится перепроверять огромное количество резюме, каждое из которых проходит первичный отбор на соответствие формальным требованиям, после чего структурируется, дополняется и отправляется на ревью технической команде.

В подавляющем большинстве случаев, в лучшем случае, сотрудниками используются табличные системы (Google docs, Microsoft Excel), что приводит ко многим ограничениям и неудобствам при дальнейшей обработке данных.

Актуальность разрабатываемой системы обусловлена в первую очередь тем, что предпринимательская деятельность в секторе информационных технологий является новым видом деятельности. Отсюда вытекает неопытность некоторых предприятий, которая проявляется и при найме сотрудников. Разрабатываемая система призвана решить некоторые трудности кадрового отдела.

Целью курсовой работы является разработка информационной системы (ИС) для учета, хранения и обработки резюме кандидатов для кадрового отдела небольшой информационно-технологической (ИТ) компании. Для этого были поставлены следующие **задачи**:

1. Изучить предметную область кадровой службы.
2. Выработать требования к ИС в виде набора основных функций ИС.
3. Создать общий дизайн ИС в виде клиент-серверного распределения программного комплекса.
4. Разработать реляционная база данных.
5. Реализовать функции ИС.
6. Протестировать ИС.

К проектируемой системе были предъявлены следующие требования:

1. Основная рабочая область — сводная таблица с именами кандидатов, их контактами и статусом. В ней должны быть следующие поля:
   - ИМЯ - ФИО, либо сокращенное имя, STRING (до 256 символов)
   - email - контактный адрес электронной почты, STRING (до 256 символов)
   - позиция - тип вакансии (справочник, который задается отдельно администратором)
   - уровень - intern, junior, middle, senior, na (выбор из справочника)

    - Дата собеседования - дата
    - Решение - назначено собеседование, отказ, одобрен (выбор из справочника)
2. Возможность быстрого добавления, редактирования, удаления резюме
3. Возможность добавления, удаления, редактирования справочных записей
4. Возможность скачать полное резюме в формате pdf
5. WYSIWYG редактор с возможностью выделения текста.

1. Выбор инструментов

В качестве инструментов для разработки (technology stack) были выбраны следующие фреймворки, библиотеки и технологии:

- Laravel 8.0
- php 8.0
- Vue.js 2
- Bootstrap 4
- MySql

Теоретические основы

Анализ представленных требований заказчика и предмета приводит к заключению, что создаваемая программная система будет представлять собой информационную систему.

Согласно [4], Информационная система (ИС) — система, предназначенная для хранения, поиска и обработки информации, и соответствующие организационные ресурсы (человеческие, технические, финансовые и т. д.), которые обеспечивают и распространяют информацию.

Согласно требованиям заказчика, система должна:

- Хранить данные;
- Обрабатывать данные;
- Обеспечивать добавление, редактирование и удаление данных,

Можно прийти к выводу, что разрабатываемая информационная система должна соответствовать клиент-серверной архитектуре [5], диаграмма представлена на рис. номер 2

Выбранные инструменты, а именно, Laravel и Vue.js помогут создать надежную систему с реактивными элементами. Laravel обеспечивает корректную обработку запросов на стороне сервера, поддерживая Model-View-Controller (MVC) архитектуру. Vue.js помогает достичь реактивное поведение, поддерживая компонентную архитектуру. В совокупности вышеперечисленные возможности приведут к уменьшению необходимого количества времени для разработки ИС, уменьшению дублирования кода, и увеличению надежности.

Реализация информационной системы

2. Проектирование базы данных

В ходе анализа требований заказчика к хранимым данным, была спроектирована база данных.

2.1. Логическая и физическая модели

Логическая модель проектируемой базы данных представлена на рис. номер 1. Отношения между таблицами один к одному, что следует из того, что: а) каждый кандидат может занимать только одну позицию (php, devops); б) каждый кандидат может быть на одном уровне (junior, middle, senior); в) по каждому кандидату может быть только одно решение (назначено собеседование, отказ).

Физическая модель проектируемой базы данных представлена на рис. номер 3.

2.2. Проверка соответствия нормальным формам

Поскольку структура таблиц decisions, positions и levels одинакова, то все представленные таблицы можно разбить на 2 группы: справочные материалы - decisions, positions, levels; и таблица briefs. Поэтому, достаточно проверить на соответствие нормальным формам (НФ) по одному представителю из каждой группы.

1. Справочные материалы (decisions):
    - Все значения атомарны — 1 НФ
    - Ключ состоит только из одного атрибута id — 2 НФ
    - Отсутствуют транзитивные зависимости — 3 НФ
2. Briefs:
    - Все значения атомарны — 1 НФ
    - Ключ состоит только из одного атрибута id — 2 НФ
    - Отсутствуют транзитивные зависимости — 3 НФ

Таким образом, спроектированная база данных соответствует третьей нормальной форме, значит, дальнейшая расработка ИС может быть продолжена.

3. Создание миграций

Миграции — это способ определения, разметки базы данных. В него входят: создание таблиц, задание полей, связывания внешними ключами. С помощью Laravel миграция для каждой таблицы описывается в классе php. С помощью команды `php artisan make:migration table_name` можно сгенерировать миграцию для таблицы *table_ name*. Внутри сгенерированного класса находятся два метода: *up* и *down*.

Метод *up* запускается всякий раз, когда запускается процесс миграции, в нем описываются создаваемая таблица, ее поля и связи с другими таблицами.

Метод *down* запускается всякий раз, когда запускается процесс отката миграций, он удаляет все данные из таблицы. Откат миграций запускается следующей командой: `php artisan db:rollback`.

В листингах номер 1 и 2 продемонстрированы миграции для создания таблиц positions и briefs. В методе *up* также можно указать действие при удалении связанных значений, в данном случае произойдет каскадом, то есть все связанные записи будут удалены.

4. Создание сидеров

Сидеры в общем смысле являются первичным наполнением базы данных. Это необходимо для нормального функционирования системы. Создаются с помощью следующей команды: `php artisan make:seeder TableSeeder`

В Laravel сидеры описываются в php классе, который имеет метод *run*. Он описывает какие данные нужно внести в указанную таблицу.

В листинге номер 3 приводится пример класса-сидера для таблицы briefs.

Запустить сидеры можно с помощью команды `php artisan db:seed`.

5. Первичная настройка контроллеров

Laravel реализует архитектуру MVC (Model View Controller) и имеет ORM (Object-Relational Mapping), контроллеры обеспечивают маршрутизацию и передачу информации из модели в представление, модель представляется в виде php класса и связана с таблицей в базе данных. Также, Laravel предоставляет возможность создавать ресурсные контроллеры, которые связаны со своей моделью. Внутри себя он содержит базовые методы и маршруты, они перечислены в табл. номер 1.

Создать ресурсный контроллер можно с помощью команды `php artisan make:controller Resource --resource`.

В листинге номер 4 приведен пример ресурсного контроллера для ресурса desicions. Рассмотрим его методы:

- метод *index* вызывается при запросе вида <hostname>/<resource>, в данном случае, если опустить часть с доменом или адресом сервера, получим /decisions/. Этот метод перенаправляет на главную страницу ресурса.

- метод *create* вызывается всякий раз, когда требуется форма для создания новой записи для текущего ресурса.

- метод *store* вызывается при запросе на сохранение данных. В нем происходит обработка данных, валидация, а затем, через метод *save*, данные сохраняются в базу данных.
- метод *show* нужен для отображения конкретной записи.
- метод *edit* вызывается, когда требуется форма для редактирования записи.
- метод *update* вызывается, когда требуется обновить запись, то есть после ее редактирования и подтверждения изменений.
- метод *destroy* вызывается при удалении записи.

Заключение

В результате разработана информационная система (ИС). Для этого решены следующие задачи:

1. Изучена предметная область кадровой службы.
2. Разработаны требования к ИС в виде набора основных функций ИС.
3. Создан общий дизайн ИС в виде клиент-серверного распределения программного комплекса.
4. Разработана реляционная база данных.
5. Реализованы основные функции ИС, включая интерфейсы пользователя и выгрузку данных в pdf формате.
6. ИС протестирована на небольшом объеме произвольных данных.

Тестирование показало, что разработанная ИС удовлетворяет всем функциональным требованиям заказчика. Использование технологий, основанных на Vue.js, Laravel, и связанных с ними позволило достаточно продуктивно производить реализацию не только основных, но и дополнительных функций ИС.

Дальнейшее совершенствование ИС предлагается вести в направлении более тесной интеграции в области HR-отделов, что даст прирост производительности в отделе кадров во время процесса активного поиска и найма сотрудников.

Приложения

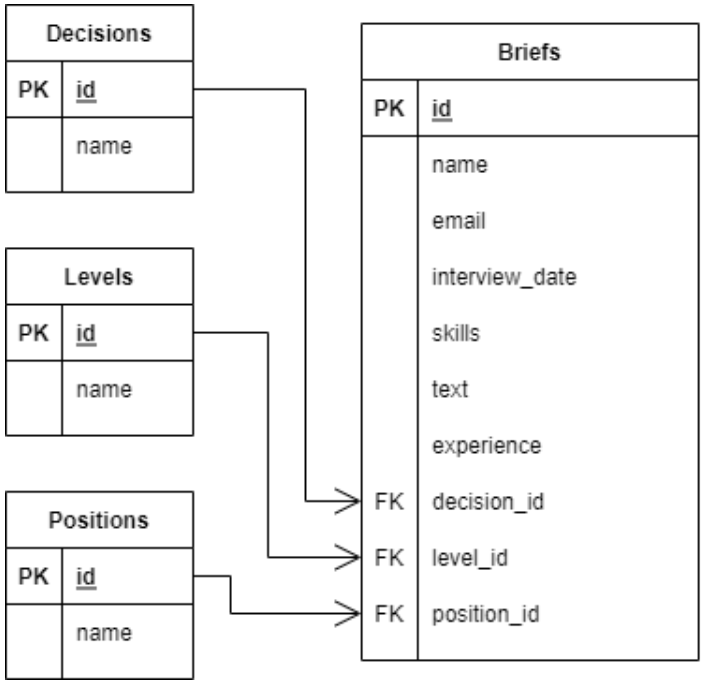

Рис. 1: Логическая модель базы данных

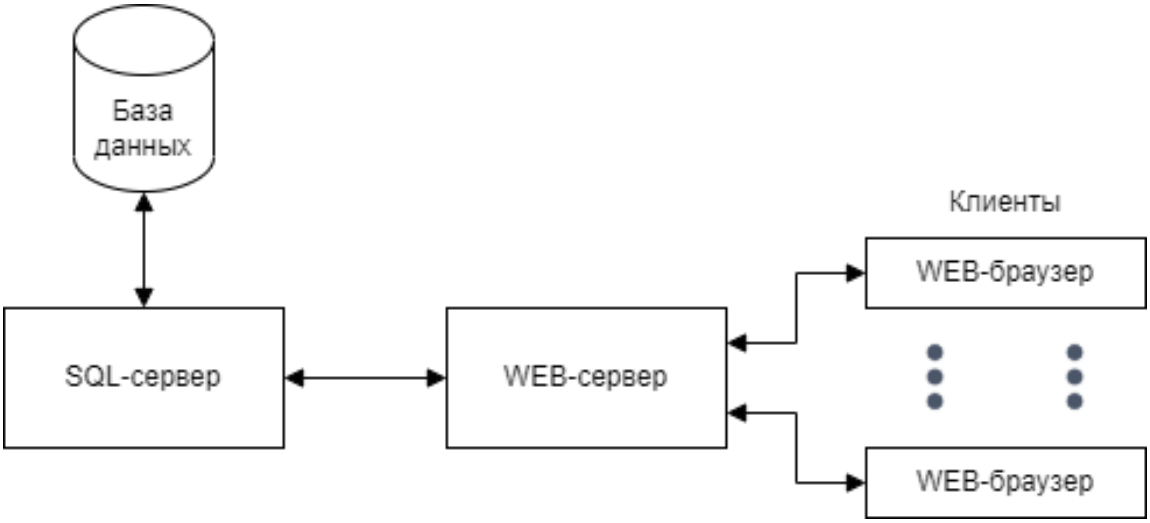

Рис. 2: Диаграмма информационной системы

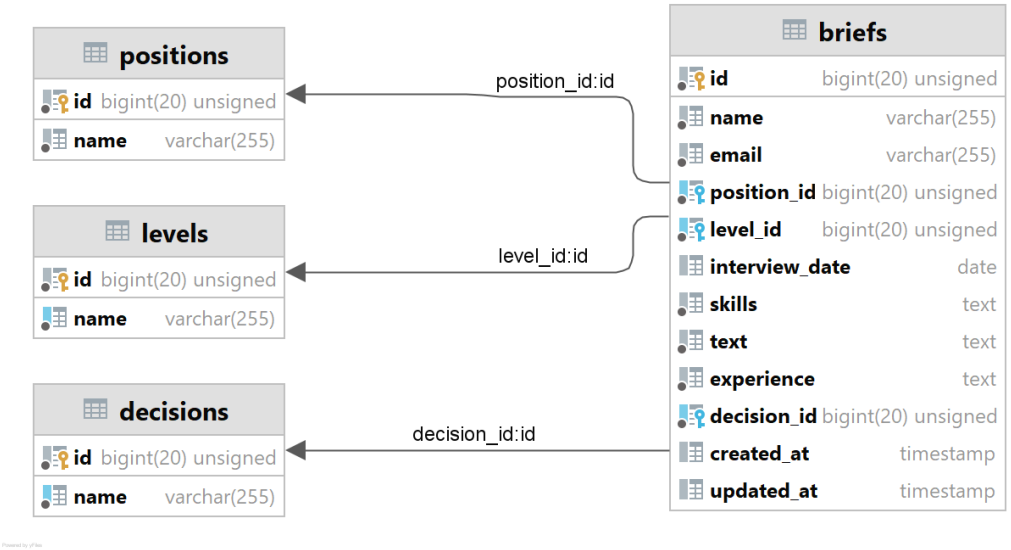

Рис. 3: Физическая модель базы данных

```
class CreatePositionsTable extends Migration
{
    /**
     * Run the migrations.
     *
```

```
     * @return void
     */
    public function up()
    {
        Schema::create('positions', function (Blueprint $table) {
            $table->id();
            $table->string("name", 255)->unique;
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('positions');
    }
}
```

Листинг 1: Класс миграции таблицы positions

```
class CreateBriefsTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('briefs', function (Blueprint $table) {
            $table->id();
            $table->string("name", 255);
            $table->string("email", 255);
            $table->foreignId("position_id")
                    ->references("id")
                    ->on("positions")
                    ->onDelete("cascade");
            $table->foreignId("level_id")
                    ->references("id")
                    ->on("levels")
                    ->onDelete("cascade");
            $table->date("interview_date")->nullable();
            $table->text("skills");
```

```
            $table->text("text");
            $table->text("experience");
            $table->foreignId("decision_id")
                    ->references("id")
                    ->on("decisions")
                    ->onDelete("cascade");
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('briefs');
    }
}
```

Листинг 2: Класс миграции таблицы briefs

```
class BriefSeeder extends Seeder
{
    /**
     * Run the database seeds.
     *
     * @return void
     */
    public function run()
    {
        $kBrief = new Brief();
        $kBrief->name = "Белогуб Константин Евгеньевич";
        $kBrief->email = "kobelogub@gmail.com";
        $kBrief->position_id = 1;
        $kBrief->level_id = 1;
        $kBrief->decision_id = 1;
        $kBrief->skills = "<ul><li>php 8</li><li>Laravel 8</li><li>HTML5, CSS3,
        ↪  JS</li><li>MySQL, PostgreSQL</li><li>IDE - PhpStorm</li></ul>";
```

```
        $kBrief->text = "<p>Я студент 3 курса бакалавриата ИМИТ ИГУ. Из учебных
        ↪  дисциплин особый интерес представили предметы, связанные с
        ↪  веб-разработкой. С php был знаком за рамками университетского
        ↪  образования - изучал самостоятельно.</p><p>Во время обучения
        ↪  довелось принять и успешно завершить курс от компании КРОК
        ↪  \"Введение в язык Java и платформу разработки</p><p>Считаю, что нет
        ↪  проблемы или задачи, с которой совсем было бы невозможно
        ↪  разобраться, наблюдал на личном опыте, но мне требуется некоторое
        ↪  время.</p><p>Обладаю аналитическим складом ума, усидчивостью и
        ↪  трудолюбием.</p>";
        $kBrief->experience = "<p>Без опыта работы.</p>";
        $kBrief->save();

        $dBrief = new Brief();
        $dBrief->name = "Разманова Дарья Константиновна";
        $dBrief->email = "razmanovad@mail.ru";
        $dBrief->position_id = 1;
        $dBrief->level_id = 1;
        $dBrief->decision_id = 1;
        $dBrief->skills = "<ul><li>HTML5, JS, CSS,
        ↪  PHP</li><li>Java</li><li>C++</li><li>Python</li><li>MySQL,
        ↪  PostgreSQL</li></ul>";
        $dBrief->text = "<p>Я студентка 3 курса ИМИТ ИГУ.</p><p> Умею работать в
        ↪  команде. Участвовала в нескольких хакатонах (разработка игр, сайт по
        ↪  отслеживанию вырубки лесов, разработка мобильных приложений). Была в
        ↪  роли как дизайнера, так и программиста. </p><p>До курса не была
        ↪  знакома с PHP. Разрабатывала игру на JS.</p><p>Знаю английский
        ↪  язык.</p><p>Закончила художественную школу с отличием.</p>";
        $dBrief->experience = "<p>Без опыта работы.</p>";
        $dBrief->save();
    }
}
```

Листинг 3: Сидер для таблицы briefs

Запрос	URI	Метод	Имя пути
GET	*/decisions*	index	decisions.index
GET	*/decisions/create*	create	decisions.create
POST	*/decisions*	store	decisions.store
GET	*/decisions/{decision}*	show	decisions.show
GET	*/decisions/{decision}/edit*	edit	decisions.edit
PUT/PATCH	*/decisions/{decision}*	update	decisions.update
DELETE	*/decisions/decision*	destroy	decisions.destroy

Таблица 1: Предоставляемые ресурсным контроллером маршруты и методы

```
    class DecisionsController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return Response
     */
    public function index()
    {
        $decisions = Decision::pluck("name", "id");
        return view('decisions.view', compact('decisions') );
    }

    /**
     * Show the form for creating a new resource.
     *
     * @return Response
     */
    public function create()
    {
        return new Response(view('decisions.create'));
    }

    /**
     * Store a newly created resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @return Response
     */
    public function store(Request $request)
    {
        //dd($request);

        $decision = new Decision;
        $request->validate([
            'new_decision' => 'bail|required',
        ]);
        $decision->name = $request->new_decision;
        $decision->save();
        return new Response(view("decisions.view")->with("decisions",
        ↪  Decision::all()));
    }

    /**
     * Display the specified resource.
```

```
     *
     * @param  \App\Models\Decision  $decision
     * @return Response
     */
    public function show(Decision $decision)
    {
        return new Response(view('decisions.show')->with("decision",
        ↪  $decision));
    }

    /**
     * Show the form for editing the specified resource.
     *
     * @param  \App\Models\Decision  $decision
     * @return Response
     */
    public function edit(Decision $decision)
    {
        return new Response(view("decisions.edit")->with("decision",
        ↪  $decision));
    }

    /**
     * Update the specified resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @param  \App\Models\Decision  $decision
     * @return Response
     */
    public function update(Request $request, Decision $decision)
    {
        $decision->update($request->all());

        return new Response();
    }

    /**
     * Remove the specified resource from storage.
     *
     * @param  \App\Models\Decision  $decision
     * @return Response
     */
    public function destroy(Decision $decision)
    {
        $decision->delete();
```

```
        return new Response();
    }
}
```

Листинг 4: Ресурсный контроллер для desicions

Листинг 5

Литература

[1] Laravel Documentation. [Электронный ресурс]. URL: https://laravel.com/docs/8.x (дата обращения: 10.10.2021).

[2] Vue.js Documentation. [Электронный ресурс]. URL: https://vuejs.org/v2/guide/ (дата обращения: 10.10.2021).

[3] Введение в системы баз данных, 8-е издание.: Пер. с англ. — М.: Издательский дом "Вильяме 2005. — 1328 с.: ил. — Парал. тит. англ

[4] Информационная система [Электронный ресурс]: Википедия. Свободная энциклопедия. – Режим доступа: https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0 (дата обращения: 10.10.2021).

[5] Клиент—сервер [Электронный ресурс]: Википедия. Свободная энциклопедия. – Режим доступа: https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B8%D0%B5%D0%BD%D1%82_%E2%80%94_%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80 (дата обращения: 10.10.2021).