

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

# Оглавление

Введение	4
Выбор инструментов	6
Выбор инструментов	6
Теоретические основы	8
Реализация информационной системы	10
Проектирование базы данных . . . . .	10
Создание миграций . . . . .	11
Создание сидеров . . . . .	11
Первичная настройка контроллеров . . . . .	11
Верстка страниц . . . . .	12
Взаимодействие с пользователем . . . . .	13
Laravel Mix . . . . .	14
Заключение	17
Список использованных источников	18
Литература	18
Приложение А	20
Приложение Б	39

# 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 реляционную базу 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.

## 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 5

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

## Проектирование базы данных

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

### Логическая и физическая модели

Логическая модель проектируемой базы данных представлена на рис. номер 4. 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 6.

### 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 разработка ИС может быть продолжена.

## 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.

## 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`.

## 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 `decisions`. 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, верификация, 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.

## Верстка страниц

Основными инструментами при верстке страниц являлись шаблонизатор `Blade` и фреймворк `Bootstrap`.

### Шаблонизатор `Blade`

Laravel предоставляет собственный шаблонизатор `blade`, он позволяет создавать шаблоны страниц, которые можно расширять другими шаблонами. В листинге номер 5 приведен код базового макета. В нем выносятся все самое общее страниц: метаданные, базовая верстка, некоторые скрипты и стили, что позволяет в дальнейшем сконцентрироваться на конкретном содержимом страниц. Блок `@yield` позволяет в расширяющем шаблоне создать секцию, которая встроится в родительский шаблон вместо этого блока.

Следующий шаблон, расширяющий базовый, приведен в листинге 6. В него было решено вынести навигационную панель и скомпилированный скрипт, чтобы не делать этого на остальных страницах.

В листинге 7 приведен код главной страницы. Вынесение одинаковых частей в родительские шаблоны значительно уменьшило количество строк в файлах, ответственных за отображение данных.

### Навигационная панель

Bootstrap предоставляет навигационную панель, она также является адаптивной под разные размеры экранов. Для удобства, навигационная панель была вынесена в отдельный Vue-компонент. Код компонента навигационной панели представлен в листинге 8. Вынесение навигационной панели позволило внедрить ее в один из родительских шаблонов (см. листинг 6).

### Страницы ввода данных

В требованиях заказчика к разрабатываемой информационной системе указано наличие WYSIWYG-редактора, поэтому, его необходимо внедрить в формы пользовательского ввода. В качестве такого редактора был выбран Quill. В листинге ?? приведен код страницы создания новой записи в таблице `briefs`. Пример отображения страницы в веб-браузере приведен на рис. 7.

Для добавления справочных записей не предусмотрены отдельные страницы, вместо них есть одно поле ввода. Такое решение было принято ввиду того, что справочные таблицы содержат только 2 поля: `id`, и `name`, поэтому, вводить нужно только название справочной записи. Пример отображения страницы в веб-браузере приведен на рис. 8.

Таким образом, был разработан пользовательский интерфейс, включающий в себя: сводную таблицу с отображением записей из таблицы `briefs` и формы для создания новых записей: как в справочных таблицах, так и в `briefs`.

### Взаимодействие с пользователем

В данной системе используется javascript-фреймворк Vue.js 2. Он позволяет создавать реактивные компоненты. В системе реактивность необходима при первичной обработке данных, и в основном

она используется на главной странице в таблице. Код компонента главной таблицы приведен в листинге 9.

Vue.js позволяет комбинировать компоненты, использовать одни внутри других. В листинге 9 приведенный компонент содержит внутри себя еще 2 компонента — `Column` и `TableRow`.

С помощью классов тегов из `Bootstrap` удалось сделать этот компонент адаптивным: некоторые колонки будут скрываться, если размеры экрана меньше некоторого (см. рис. 1, 2, 3).

Vue.js предоставляет несколько возможностей взаимодействия дочерних и родительских компонентов, рассмотрим 2 из них: `props` позволяет передавать данные из родительского компонента в дочерний, в данном случае в листинге 9 мы передаем компонентам `Column` следующие данные: `value`, `data`, `filter`, `resource`; `emit` же позволяет передавать данные из дочернего компонента в родительский посредством событий: в данном случае в компонентах `Column` могут возникнуть события `changed` и `del`, в результате которых вызываются назначенные методы из листинга 9 родительского компонента.

Можно не только оповещать о каких-либо событиях, но и передавать данные: в данном случае в результате события `changed`, родительскому компоненту могут передаться некоторые данные, а именно `JSON` объект, в котором содержатся данные о назначенных сортировках и фильтрах.

В листинге 9 используется директива `v-bind`. Она позволяет закрепить некоторые данные, в данном случае `JSON`-объект, за конкретным компонентом. Полученные данные становятся свойством компонента, аналогично параметрам, переданным через `props`. Также используется директива `v-for`. Она позволяет обходить коллекцию элементов, создавая теги для каждого: в данном случае для каждого элемента из коллекции `this.rows` будет создан тег `<div>` с вложенным компонентом `TableRow`.

Таким образом, родительский компонент отвечает за загрузку и передачу данных дочерним компонентам, которые в ответ на некоторые пользовательские действия сообщают родительскому об изменении состояния.

## Laravel Mix

Laravel предоставляет интерфейс для сборки проекта. Он компилирует все `js` и `css` файлы проекта, в итоге создается по 1 файлу, содержащие в себе все скрипты и стили.

В листниге 10 подключается Vue.js, а затем объявляются компоненты, чтобы они были доступны на всех страницах, на которых подключен скомпилированный js-файл.

## 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.

## 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 "Вильямс 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](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](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).

# Приложение А

```
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
→ РНР. 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

```

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' => '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 decisions

```

<!doctype html>
<html lang="ru">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
        content="width=device-width, user-scalable=no,
        ↪ initial-scale=1.0, maximum-scale=1.0,
        ↪ minimum-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">

```

```

<meta name="csrf-token" content="<?php echo csrf_token(); ?>"
↳ id="token">
<title>@yield('title')</title>
<link rel="stylesheet" href="/css/app.css">
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.1/
dist/css/bootstrap.min.css" rel="stylesheet"
↳ integrity="sha384-F3w7mX95PdgyTmZZMECAngseQ
B83DfGTowi0iMjiWaeVhAn4FJkqJByhZMI3AhiU"
↳ crossorigin="anonymous">
<link rel="stylesheet" href="/css/bootstrap.css">
<link rel="stylesheet" href="/css/medium-editor.css">
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.1/
dist/js/bootstrap.bundle.min.js"
↳ integrity="sha384-/bQdsTh/da6pkI1MST/
rWKFjnJaCP5gBSY4sEBT38Q/9RBh9AH40zEOg7Hlq2THRZ"
↳ crossorigin="anonymous"></script>
</head>
<body>
<div id="navigation" class="container">
@yield('navigation')
</div>
<section class="d-flex flex-column justify-content-center
↳ align-items-center">
<h1 class="mt-3 mb-3">
    @yield('title')
</h1>
<div class="container d-flex flex-column justify-content-center
↳ align-items-center">
    @yield('content')
</div>
</section>
</body>
@stack('scripts')
</html>

```

Листинг 5: Базовый шаблон

```

@extends("base")

@section('navigation')

```

```

    <v-nav></v-nav>
@endsection

@prepend('scripts')
    <script src="/js/app.js"></script>
@endprepend

```

Листинг 6: Второй базовый шаблон

```

@extends("template")
@section("title")
    Главная
@endsection

<?php
use App\Models\Brief;
/**
 * @var Brief[] $briefs
 */

?>

@section("content")
    <div class="row col-4 d-flex justify-content-around">
        <a class="btn col-auto btn-outline-dark mb-3"
            ↪ href="/briefs/create">Добавить резюме</a>
        </div>
        <div id="dashboardTable" class="container col-11 ml-0
            ↪ ml-sm-auto">
            <v-db-table columns="Имя E-mail Позиция Уровень Дата
                ↪ Решение"></v-db-table>
            </div>
    @endsection

@push('scripts')
    <script>
        const table = new Vue({
            el: "#dashboardTable",
        })
        const updateButton =
            ↪ document.getElementById('updateTableButton');
        updateButton.onclick = function () {

```

```

        table.$children[0].changes([]);
    }
</script>
@endpush

```

Листинг 7: Шаблон главной страницы

```

<template>
  <nav class="navbar navbar-expand-lg navbar-dark bg-dark
    ↳ rounded">
    <a class="navbar-brand" href="/briefs">CRM</a>
    <button class="navbar-toggler" type="button"
      ↳ data-toggle="collapse" data-target="#collapsingList"
      ↳ aria-controls="collapsingList" aria-expanded="false"
      ↳ aria-label="Toggle navigation">
      <span class="navbar-toggler-icon"></span>
    </button>

    <div class="collapse navbar-collapse" id="collapsingList">
      <div class="navbar-nav ml-auto">
        <a class="nav-item nav-link"
          ↳ href="/decisions">Решения</a>
        <a class="nav-item nav-link" href="/levels">Уровни</a>
        <a class="nav-item nav-link"
          ↳ href="/positions">Позиции</a>
      </div>
    </div>
  </nav>
</template>

<script>
export default {
  name: "Nav"
}
</script>

<style scoped>

</style>

```

Листинг 8: Компонент навигационная панель

```

<template>
  <div class="d-flex flex-column container table-bordered col-11
    ↳ col-sm-12 pl-0 pr-0 text-center" id="table">
    <div class="d-flex flex-row container col-12 pl-0 pr-0 mt-1 mb-1
      ↳ fw-bold">
      <Column value="Имя" class="col-4 col-sm-3 col-md-2 d-flex
        ↳ flex-row justify-content-around align-items-center pl-0
        ↳ pr-0"
        data="name" :filter="false" v-on:changed="changes"
          ↳ v-on:del="deleteFromSort"></Column>
      <Column value="E-mail" class="col-2 d-flex flex-row
        ↳ justify-content-around align-items-center pl-0 pr-0
        ↳ d-none d-sm-none d-md-block"
        data="email" :filter="false" v-on:changed="changes"
          ↳ v-on:del="deleteFromSort"></Column>
      <Column value="Позиция" class="col-4 col-sm-3 col-md-2
        ↳ d-flex flex-row justify-content-around align-items-center
        ↳ pl-0 pr-0"
        data="position_id" :filter="true" resource="positions"
          ↳ v-on:changed="changes"
          ↳ v-on:del="deleteFromSort"></Column>
      <Column value="Уровень" class="col-md-2 col-sm-3 d-flex
        ↳ flex-row justify-content-around align-items-center pl-0
        ↳ pr-0 d-none d-sm-block"
        data="level_id" :filter="true" resource="levels"
          ↳ v-on:changed="changes"
          ↳ v-on:del="deleteFromSort"></Column>
      <Column value="Дата" class="col-2 d-flex flex-row
        ↳ justify-content-around align-items-center pl-0 pr-0
        ↳ d-none d-sm-none d-md-block"
        data="interview_date" :filter="true" resource="date"
          ↳ v-on:changed="changes"
          ↳ v-on:del="deleteFromSort"></Column>
      <Column value="Решение" class="col-4 col-sm-3 col-md-2
        ↳ d-flex flex-row justify-content-around align-items-center
        ↳ pl-0 pr-0"
        data="decision_id" :filter="true" resource="decisions"
          ↳ v-on:changed="changes"
          ↳ v-on:del="deleteFromSort"></Column>
    
```

```

    </div>
    <div v-for="$data in this.rows">
      <TableRow v-bind="$data" v-on:reload="load">
        </TableRow>
      </div>
    </div>
  </template>

<script>

import TableRow from "./TableRow";
import Column from "./Column";

export default {
  name: "DashBoardTable",
  components: {Column, TableRow},
  data() {
    return {
      rows: [],
      request: window.location.origin + '/api/briefs',
      req: [],
      sorts: [],
      filters: [],
    };
  },
  props: {
    columns: String,
  },
  created() {
    this.load();
  },
  methods: {
    load(req = '') {
      console.log(this.request + "?" + req);
      axios
        .get(this.request + "?" + req)
        .then(response => (this.rows = response.data));
    },
    changes(arr) {
      let requ = [];
      if (arr.hasOwnProperty('sorts')) {

```

```

        this.sorts[arr['sorts']['data']] = arr['sorts']['name'];
        let req = Object.values(this.sorts).join();
        if (req === ""){
            this.req['sort'] = "";
        } else {
            this.req['sort'] = "sort=" + req;
        }
    }
    if (arr.hasOwnProperty('filters')) {
        if (!arr['filters']['data'].length){
            delete this.filters[arr['filters']['resource']]
        } else {
            this.filters[arr['filters']['resource']] =
                ↪ arr['filters']['data'];
        }
        let req = [];
        for (let filter in this.filters){
            req.push(filter + ":" + this.filters[filter].join("|"));
        }
        if (req.length) {
            this.req['filter'] = "filter=" + req.join();
        } else {
            this.req['filter'] = ' ';
        }
    }
    this.load(Object.values(this.req).join("&"));
},
deleteFromSort(name) {
    delete this.sorts[name];
    let req = Object.values(this.req).join("&");
    console.log(req);
    this.load(req);
},
},
}
</script>
<style scoped>

```

---

</style>

---

Листинг 9: Компонент главная таблица

```

window.Vue = require('vue').default;

import Vue2TouchEvents from 'vue2-touch-events';
Vue.use(Vue2TouchEvents);

// const files = require.context( './', true, /\.vue$/i)
// files.keys().map(key =>
  ↳ Vue.component(key.split( '/' ).pop().split( '.' )[0],
  ↳ files(key).default))

Vue.component('v-db-table',
  ↳ require('./components/dashboard/DashBoardTable.vue').default);
Vue.component('v-nav', require("./components/Nav.vue").default);
Vue.component('v-tool-table',
  ↳ require('./components/tools/ToolTable.vue').default);

const navigation = new Vue({
  el: '#navigation',
});

```

Листинг 10: Главный скриптовый файл

```

@extends("template")

@section("title")
    Добавить резюме
@endsection

<?php
use App\Http\Controllers\BriefsController;
use App\Models\Brief;use App\Models\Decision;
use App\Models\Level;
use App\Models\Position;
use Carbon\Carbon;
?>
@section("content")

```



```

    {{Form::open(["action" =>
→   'App\Http\Controllers\BriefsController@store', "method" =>
→   "POST", "class" => "container"])}}
    @csrf
    <div class="form-group">

        @include('errors')

        <label for="name">ФИО</label>
        {{Form::text("name", null, ["placeholder" => "ФИО", "class"
→   => "form-control", "id"=>"name"])}}

    </div>

    <div class="form-group">
        <label for="position_id">Позиция</label>
        {{Form::select("position_id", $positions, null, ["class" =>
→   "form-control", "id"=>"position_id"])}}
    </div>
    <div class="form-group">
        <label for="email">E-mail</label>
        {{Form::text("email", null, ["placeholder" => "E-mail", "class"
→   => "form-control", "id"=>"email"])}}
    </div>

    <div class="form-group">
        <label for="level_id">Уровень</label>
        {{Form::select("level_id", $levels, null, ["class" =>
→   "form-control", "id"=>"level_id"])}}
    </div>
    <div class="form-group">
        <label for="decision_id">Решение</label>
        {{Form::select("decision_id", $decisions, null, ["class" =>
→   "form-control", "id"=>"decision_id"])}}
    </div>
    <div class="form-group">
        <label for="interview_date">Дата собеседования</label>
        {{Form::date("interview_date", null, ["class" => "form-control",
→   "id"=>"interview_date"])}}
    </div>
    <div class="form-group">

```

```

    <label for="skills">Ключевые навыки</label>
    {{Form::textarea("skills", null, ["placeholder" => "Ключевые
↳ навыки", "class" => "form-control", "id"=>"skills",
↳ "style"=>"display: none"])}}
    <div id="quill-skills"></div>
  </div>
  <div class="form-group editor">
    <label for="experience">Опыт работы</label>
    {{Form::textarea("experience", null, ["placeholder" => "Опыт
↳ работы", "class" => "form-control", "id"=>"experience",
↳ "style"=>"display: none"])}}
    <div id="quill-exp"></div>
  </div>
  <div class="form-group editor">
    <label for="text">Резюме</label>
    {{Form::textarea("text", null, ["placeholder" => "Резюме",
↳ "class" => "form-control", "id"=>"text", "style"=>"display:
↳ none"])}}
    <div id="quill-text"></div>
  </div>
  <div class="form-group">
    {{Form::submit("Отправить", ["class" => "btn btn-submit
↳ form-control btn-outline-dark", "id" => "submit"])}}
  </div>
  <div id="editor"></div>
  {{Form::close()}}
@endsection

@prepend('scripts')
<script>
  $(document).on('change', '#name', createEmail);
  $(document).on('change', '#position_id', createEmail);
  function createEmail(){

    var all_email = '<?php echo Brief::pluck("email");?>';
    //console.log(all_email);

    var arr = $('#name').val().split(' ');
    if (arr.length === 1) {
      var name_for_email = translit(arr[0]);
    }
  }

```

```

        else {
            var name_for_email = translit(arr[0]) + "." +
↪ translit(arr[1]);
        }
        var pos = "-" + $(' #position_id
↪ option:selected').text().substr(0,3) + "@adict.ru";

        while (all_email.indexOf(name_for_email+pos) !== -1) {
            name_for_email += Math.round(Math.random()*10);
        }
        var val_email = name_for_email + pos;

        $(' #email').val(val_email);
    }

function translit(word) {
    var converter = {
        'a': 'a', 'б': 'b', 'в': 'v', 'г': 'g', 'д': 'd',
        'e': 'e', 'ё': 'e', 'ж': 'zh', 'з': 'z', 'и': 'i',
        'й': 'y', 'к': 'k', 'л': 'l', 'м': 'm', 'н': 'n',
        'о': 'o', 'п': 'p', 'р': 'r', 'с': 's', 'т': 't',
        'у': 'u', 'ф': 'f', 'х': 'h', 'ц': 'c', 'ч': 'ch',
        'ш': 'sh', 'щ': 'sch', 'ь': '', 'ы': 'y', 'ъ': '',
        'э': 'e', 'ю': 'yu', 'я': 'ya'
    };

    word = word.toLowerCase();

    var answer = '';
    for (var i = 0; i < word.length; ++i) {
        if (converter[word[i]] === undefined) {
            answer += word[i];
        } else {
            answer += converter[word[i]];
        }
    }

    answer = answer.replace(/[^0-9a-z]/g, '-');
    answer = answer.replace(/[-]/g, '-');
    answer = answer.replace(/^\-|-$/g, '');
    return answer;

```

```

    }
</script>
<link href="https://cdn.quilljs.com/1.3.6/quill.snow.css"
↪ rel="stylesheet">
<script src="https://cdn.quilljs.com/1.3.6/quill.js"></script>
<script>
    var skillsTA = document.getElementById("skills");
    var expTA = document.getElementById("experience");
    var textTA = document.getElementById("text");

    var skills = new Quill("#quill-skills", {
        theme: 'snow',
        modules: {
            toolbar: [['bold', 'italic', 'underline', 'link'], [{ 'list':
↪ 'ordered'}, { 'list': 'bullet' }]]
        },
        placeholder: "Ключевые навыки",
    });
    var exp = new Quill("#quill-exp", {
        theme: 'snow',
        modules: {
            toolbar: [['bold', 'italic', 'underline', 'link'], [{ 'list':
↪ 'ordered'}, { 'list': 'bullet' }]]
        },
        placeholder: "Опыт работы",
    });
    var text = new Quill("#quill-text", {
        theme: 'snow',
        modules: {
            toolbar: [['bold', 'italic', 'underline', 'link'], [{ 'list':
↪ 'ordered'}, { 'list': 'bullet' }]]
        },
        placeholder: "Резюме",
    })

    skills.on('text-change', function(){
        skillsTA.innerHTML = skills.root.innerHTML;
    });
    exp.on('text-change', function(){
        expTA.innerHTML = exp.root.innerHTML;
    });

```

```
text.on('text-change', function(){
    textTA.innerHTML = text.root.innerHTML;
});
text.root.innerHTML = textTA.innerHTML;
exp.root.innerHTML = expTA.innerHTML;
skills.root.innerHTML = skillsTA.innerHTML;
</script>
<style>
    .ql-editor{
        min-height: 250px!important;
    }
</style>
@endprepend
```

Листинг 11: Страница создания записи резюме

Листинг 12

## Приложение Б

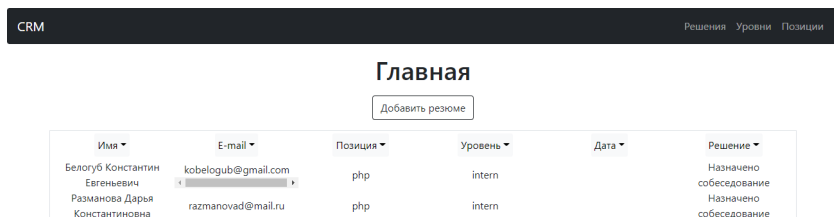


Рис. 1. Отображение главной страницы на больших экранах

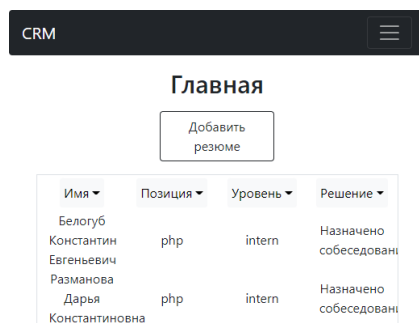


Рис. 2. Отображение главной страницы на средних экранах

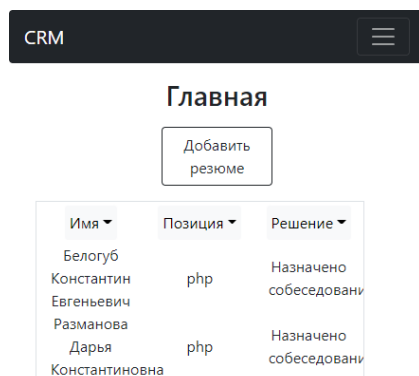


Рис. 3. Отображение главной страницы на маленьких экранах

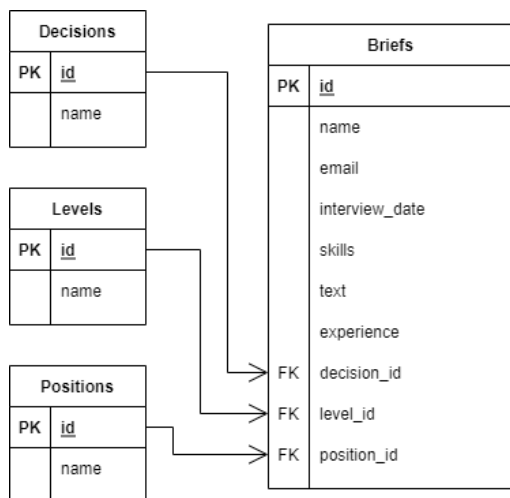


Рис. 4. Логическая модель базы данных

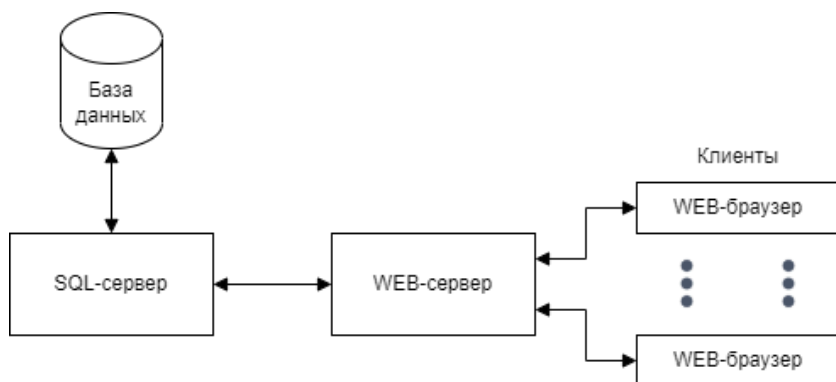


Рис. 5. Архитектура «клиент-сервер»



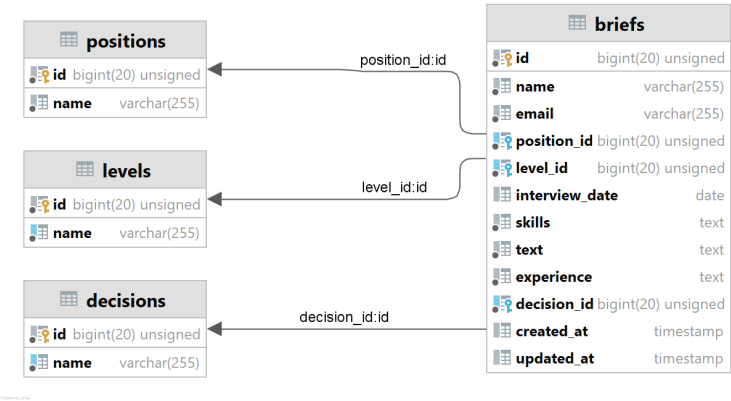


Рис. 6. Физическая модель базы данных

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

CRM

РешенияУровниПозиции

## Добавить резюме

ФИО

Белогуб Константин Евгеньевич

Позиция

php

E-mail

belogub.konstantin-php@adict.ru

Уровень

intern



Решение

Назначено собеседование

Дата собеседования



ДД.ММ.ГГГГ

Ключевые навыки

**B I U**  



- php 8
- Laravel 8
- HTML5, CSS3, JS
- MySQL, PostgreSQL
- IDE - PhpStorm

Опыт работы

**B I U**  

Без опыта работы.

Резюме

**B I U**  

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.

Во время обучения довелось принять и успешно завершить курс от компании КРОК "Введение в язык Java и платформу разработки". Считаю, что нет проблемы или задачи, с которой совсем было бы невозможно разобраться, наблюдал на личном опыте, но мне требуется некоторое время.

Обладаю аналитическим складом ума, усидчивостью и трудолюбием.

Отправить

Рис. 7. Форма создания резюме

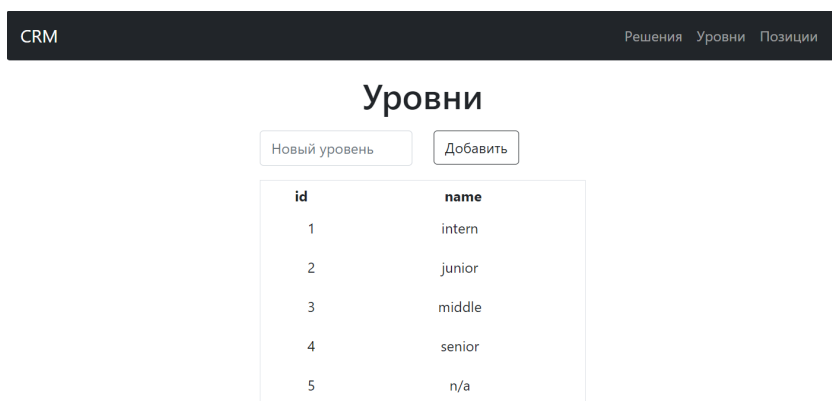


Рис. 8. Страница справочных записей