

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

Студентки 3 курса группы 2361
Размановой Дарьи Константиновны
Направление : 09.03.03 – Прикладная
информатика

Руководитель:
канд. техн. наук доцент
Черкашин Евгений Александрович

Курсовая работа защищена с оценкой

Иркутск – 2021

Оглавление

1.	Введение	3
2.	Выбор инструментов	4
3.	Теоретические основы	5
4.	Создание контроллеров	7
5.	Валидация	7
6.	Генерация e-mail	8
7.	Пользовательский интерфейс	8
8.	Заключение	11

Введение

1. 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.

Цель курсовой работы является разработка CRM-системы для учета, хранения и обработки резюме кандидатов для кадрового отдела небольшой IT-компании. Для этого были поставлены следующие задачи:

- Изучить предметную область кадровой службы.
- Выработать требования к CRM-системе в виде набора основных функций ИС.
- Создать общий дизайн CRM-системе в виде клиент-серверного распределения программного комплекса.
- Разработать реляционную базу данных.
- Реализовать функции CRM-системе.
- Протестировать CRM-системе.

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.

2. 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

3. Теоретические основы

CRM-система — это система, предназначенная для хранения, поиска и обработки информации. С ней удобно работать человеку, нанимающему сотрудников.

Со стороны заказчика были предъявлены следующие требования:

- Хранить данные;
- Обработать данные (сортировать, фильтровать);
- Обеспечивать добавление новых, редактирование и удаление данных,

Для удобной работы с данными необходимо создать пользовательский интерфейс, отвечающий всем требованиям. Разрабатываемая CRM-система должна быть понятна пользователю, то есть не содержать большое количество ненужных кнопок. На рис. 1 представлен скрин главной страницы со всеми кнопками, появляющимися только при наведении на конкретного кандидата.

Для разработки были выбраны Laravel и Vue.js, так как с их помощью можно реализовать все требования. Laravel - это бесплатный веб-фреймворк с открытым кодом, предназначенный для разработки с использованием архитектурной модели MVC (Model-View-Controller). Данный фреймворк позволит без особого труда создать модели, контроллеры и провайдеры для работы с данными. С помощью Vue.js реализуется поведение, соответствующее поставленной задаче. Vue.js может функционировать как веб-фреймворк для разработки одностраничных приложений в реактивном стиле. Это необходимо в создаваемой системе.

Реализация информационной системы

4. Создание контроллеров

Для передачи данных в базу данных, необходимо создать объект. Laravel реализует архитектуру MVC, как уже было сказано ранее. Контроллеры обеспечивают маршрутизацию и передачу информации из модели в представление. Например, контроллер `BriefsController` служит для работы с моделью `Brief`. Класс представлен на листинге 1.

В данном классе реализованы следующие методы:

- метод *index* перенаправляет на главную страницу ресурса.
- метод *create* вызывается каждый раз, при попытке создать новую запись класса `Brief` в базу данных `briefs`.
- метод *store* вызывается при запросе на сохранение данных класса `Brief`. В нем обрабатываются полученные данные, происходит их валидация, и далее, через метод *save*, данные сохраняются в базу данных `briefs`.
- метод *show* нужен для отображения конкретной записи из базы данных `briefs`.
- метод *edit* - для редактирования записи из базы данных `briefs`.
- метод *update* вызывается, при попытке обновления записи.
- метод *destroy* вызывается при удалении записи из бд `briefs`.
- метод *download* для возможности скачивания документа с резюме конкретной записи в формате PDF.

5. Валидация

Для проверки на ввод корректных данных используется валидация. В данном приложении происходит проверка данных при попытке создания и редактирования записи. Для записи нового кандидата (экземпляр класса `Brief`) проверка происходит в методе *store*, представленном на листинге 2 при сохранении данных, а также в методе *update* (листинг 3). При валидации

проверка идет слева направо, то есть для поля **position_id** сначала будет установлена остановка выполнения правил проверки после первой ошибки (bail), а затем проверка на непустоту (required).

Если какое-то поле не прошло проверку, то отображается соответствующее предупреждение, представленное в листинге 2, только после прохождения всех проверок, данные кандидата могут быть сохранены в базу данных с помощью **\$brief->save()**.

6. Генерация e-mail

Для создания уникальной почты кандидата используется его имя, должность и суффикс. При изменении имени (ФИО) или должности, автоматически происходит создание нового уникального адреса электронной почты. На листинге 4 представлен код для генерации e-mail, при создании нового кандидата.

В данном коде реализованы следующие методы:

- метод *createEmail* вызывается при изменении содержимого поля с именем или должностью кандидата. В данном методе происходит разбиение на имя, фамилию и отчество, далее перевод фамилии и имени на английский язык. Добавление 3-х букв из должности, далее проверка на уникальность созданного адреса e-mail.
- метод *translit* переводит переданное слово с русского на английский язык.

7. Пользовательский интерфейс

Существуют 4 страницы.

- Основная страница, на которой происходит взаимодействие с резюме кандидатов, там можно добавить новое резюме и редактировать или удалять имеющиеся, соответствующие кнопки всплывают при наведении на конкретную запись. Таблицу можно сортировать и использовать фильтры по значениям, инструменты для этого находятся в выпадающих списках - названиях колонок, данные для фильтрации хранятся в справочниках, о них - далее.
- Уровни, позиции, решения - страницы, на которых данные берутся из справочников, они предназначены для администратора, там можно удалять и добавлять новые справочные записи, аналогично dashboard, для этого предусмотрены всплывающие кнопки, а чтобы добавить новую запись, нужно просто ввести имя записи сверху страницы.

В данной системе для создания компонентов используется javascript фреймворк Vue.js 2. Он предоставляет удобное взаимодействия дочерних и родительских компонентов. В приложении таблица на главном экране, также как и таблицы справочников состоят из строк и колонок. Это дочерние элементы: **Column** и **Row**, которые получают из таблицы необходимые данные. Рассмотрим эти компоненты подробнее.

1. Компонент **Column** Компонент **Column** получает свойства props: value, data, filter, resource.

-
- Переменная `resource` отвечает за фильтрацию колонок в таблице. Если же данная переменная пустая, то фильтрация данной колонки будет отсутствовать.
 - В методе `created` для всех колонок, кроме колонки с датой, получаем путь для считывания данных. Для `resource === "date"` существует особое поведение, так как там могут быть нулевые или повторяющиеся значения. Далее делается запрос к ресурсу. После получения ответа массив состояния `filterOptions` заполняется полученными данными.
 - В методе `clickSort` выбирается тип сортировки (по убыванию или по возрастанию). Далее через `emit` происходит передача информации в родительский компонент.
2. Компонент **TableRow** Компонент `TableRow` получает данные для каждой строки из родительского компонента. Они записываются в `props`.
В методе `data` указывается начальное состояние строки.
При клике на конкретную строку происходит переход на страницу с данными кандидата.
При наведении курсора мыши на конкретного кандидата появляются кнопки для удаления, редактирования и скачивания данных.

Заключение

8. Заключение

В результате разработана CRM-система. При реализации данной системы были решены следующие задачи:

1. Создан удобный интерфейс для работы в системе.
2. Перед записью все данные из полей проходят проверку.
3. Происходит генерация уникального адреса e-mail.

В результате проделанной работы совместно с Белогубом Константином было создано работающее приложение. В данной CRM-системе можно работать с данными кандидата, а именно, добавлять, редактировать и сохранять резюме выбранного кандидата в формате PDF. Также возможно удаление данных как самого пользователя, так и записей из справочников с должностями и уровнями.

Далее планируется создание регистрации пользователя с помощью Laravel, и тем самым предоставление работы с данными справочников только администраторам.

Приложения

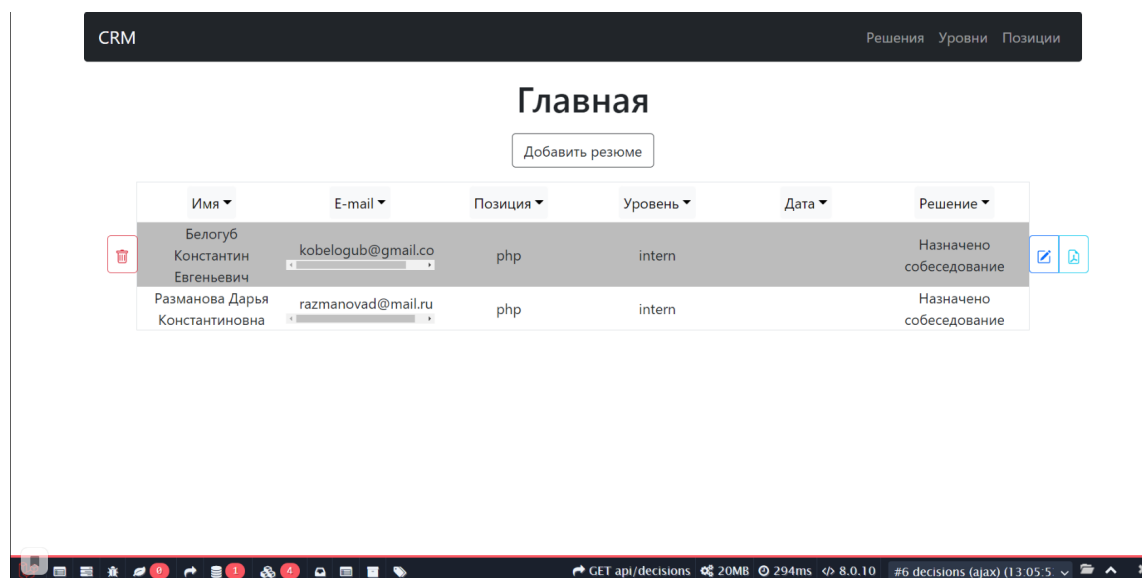


Рис. 1: Наведение курсора на кандидата

CRM

РешенияУровниПозиции

Добавить резюме

ФИО

ФИО

Позиция

php

E-mail

E-mail

Уровень

intern




Решение

Назначено собеседование

Дата собеседования




дд.мм.гггг

Ключевые навыки

B **I** **U**   




Ключевые навыки

Опыт работы

B **I** **U**   

Опыт работы

Резюме

B **I** **U**   

Резюме

Отправить

Рис. 2: Создание нового кандидата

CRM Решения Уровни Позиции

Просмотр резюме

Имя	Разманова Дарья Константиновна
E-mail	razmanovad@mail.ru
Позиция	php
Уровень	intern
Дата	
Ключевые навыки	<ul style="list-style-type: none"> • HTML5, JS, CSS, PHP • Java • C++ <div> <ul style="list-style-type: none"> • HTML5, JS, CSS, PHP • Java • C++ • Python • MySQL, PostgreSQL </div>
Опыт	Без опыта работы.
Резюме	<p>Я студентка 3 курса ИМИТ ИГУ.</p> <p>Умею работать в команде. Участвовала в нескольких хакатонах (разработка игр</p> <div> <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> </div>

Редактировать
PDF
Удалить

Рис. 3: Просмотр информации кандидата

```

class BriefsController extends Controller
{
    public function index(): Response
    {
        return new Response(view("briefs.view"));
    }
}

```

```
public function create(): Response
{
    ...
}

public function store(Request $request)
{
    $brief = new Brief();
    ...
    return redirect('/briefs/');
}

public function show(Brief $brief): Response
{
    return new Response(view("briefs.show")->with("brief", $brief));
}

public function edit(Brief $brief): Response
{
    ...
}

public function update(Request $request, Brief $brief)
{
    ...
    $brief->update($request->all());
    return redirect("briefs/$brief->id");
}

public function destroy(Brief $brief): Response
{
    $brief->delete();
    return new Response(redirect("briefs"));
}

public function download(Brief $brief)
{
    $name = str_replace(" ", "_", $brief->name) . "_" .
        ↳ strtoupper($brief->position->name);
    $pdf = PDF::loadView('briefs.topdf', compact('brief'));
    return $pdf->download("$name.pdf");
}
}
```

Листинг 1: Класс контроллера BriefsController


```
public function store(Request $request)
{
    $brief = new Brief();
    $values = $request->all([
        "name",
        "position_id",
        "email",
        "level_id",
        "interview_date",
        "skills",
        "text",
        "experience",
        "decision_id",
    ]);
    $brief->fill($values);
    $request->validate([
        'name' => 'bail|required|max:255',
        'position_id' => 'bail|required',
        'email' => 'bail|required|email:filter|max:255|unique:briefs',
        'level_id' => 'bail|required',
        'skills' => 'bail|required|max:2000',
        'text' => 'bail|required|max:8000',
        'experience' => 'bail|required|max:10000',
        'decision_id' => 'bail|required',
    ], [
        'name.required' => 'Заполните ФИО',
        'position_id.required' => 'Заполните поле позиция',
        'email.required' => 'Заполните поле email',
        'level_id.required' => 'Заполните поле уровень',
        'skills.required' => 'Заполните поле навыки',
        'experience.required' => 'Заполните поле опыт',
        'decision_id.required' => 'Заполните поле решение',
        'text.required' => 'Заполните поле резюме',

        'email.unique' => 'Кандидат с таким именем уже существует! Введите
        ↳ другой email',

        'email.email' => 'Введите корректный email',
        'email.max' => 'Кол-во символов в EMAIL не более 255',
        'skills.max' => 'Кол-во символов в КЛЮЧЕВЫХ НАВЫКАХ не более 2000',
        'text.max' => 'Кол-во символов в РЕЗЮМЕ не более 8000',
        'experience.max' => 'Кол-во символов в ОПЫТЕ не более 10000',
    ]);
    $brief->save();
}
```

```

    return redirect('/briefs/');
}

```

Листинг 2: Метод store из класса BriefsController

```

public function update(Request $request, Brief $brief)
{
    $request->validate([
        'name' => 'bail|required|max:255',
        'position_id' => 'bail|required',
        'email' =>
            ↳ 'bail|required|email:filter|max:255|unique:briefs,id,'.$brief->id,
        'level_id' => 'bail|required',
        'skills' => 'bail|required|max:2000',
        'text' => 'bail|required|max:8000',
        'experience' => 'bail|required|max:10000',
        'decision_id' => 'bail|required',
    ], [
        'name.required' => 'Заполните ФИО',
        'position_id.required' => 'Заполните поле позиция',
        'level_id.required' => 'Заполните поле уровень',
        'skills.required' => 'Заполните поле навыки',
        'experience.required' => 'Заполните поле опыт',
        'decision_id.required' => 'Заполните поле решение',
        'text.required' => 'Заполните поле резюме',

        'email.unique' => 'Кандидат с таким именем уже существует! Введите
            ↳ другой email',
        'email.email' => 'Введите корректный email',
        'email.max' => 'Кол-во символов в EMAIL не более 255',
        'skills.max' => 'Кол-во символов в КЛЮЧЕВЫХ НАВЫКАХ не более 2000',
        'text.max' => 'Кол-во символов в РЕЗЮМЕ не более 8000',
        'experience.max' => 'Кол-во символов в ОПЫТЕ не более 10000',
    ]);
    $brief->update($request->all());
    return redirect("briefs/$brief->id");
}

```

Листинг 3: Метод update из класса BriefsController

```

@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;
$('#email').val(val_email);
}
function translit(word) {
    var converter = {
        'а': 'a', 'б': 'b', 'в': 'v', 'г': 'g', 'д': 'd',
        'е': 'e', 'ё': 'e', 'ж': 'zh', 'з': 'z', 'и': 'i',
        'й': 'y', 'к': 'k', 'л': 'l', 'м': 'm', 'н': 'n',
        'о': 'o', 'п': 'p', 'р': 'r', 'с': 's', 'т': 't',
        'у': 'u', 'ф': 'f', 'х': 'h', 'ц': 'c', 'ч': 'ch',
        'ш': 'sh', 'щ': 'sch', 'ъ': '', 'ы': 'y', 'ь': '',
        'э': 'e', 'ю': 'yu', 'я': 'ya'
    };
    word = word.toLowerCase();
    var answer = '';
    for (var i = 0; i < word.length; ++i) {
        if (converter[word[i]] === undefined) {
            answer += word[i];
        } else {
            answer += converter[word[i]];
        }
    }
    answer = answer.replace(/[~0-9a-z]/g, '-');
    answer = answer.replace(/[-]/g, '-');
    answer = answer.replace(/^\-|-$/g, '');
    return answer;
}
</script>

```

Листинг 4: Генерация уникального e-mail

```
<script>
```

```

export default {
  name: "Column",
  props: {
    value: String,
    data: String,
    filter: Boolean,
    resource: String,
  },
  created() {
    if (this.resource !== "date" && this.resource){
      axios
        .get(window.location.origin + "/api/" + this.resource)
        .then(response => this.filterOptions = response.data);
    }
    else if (this.resource === "date"){
      axios
        .get(window.location.origin + "/api/briefs-dates")
        .then(response => {
          this.filterOptions = response.data;
          this.filterOptions = [...new Set(this.filterOptions)]
          this.filterOptions = this.filterOptions.map(function(el,
            ↪ index, arr){
            return {id: el, name: ( el ? el : "Не указана")});
          })
        });
    }
  },
  methods: {
    clickSort() {
      this.asc = !this.asc;
      this.ascText = this.asc ? "DESC" : "ASC";
      this.$emit('changed', {'sorts' : {'data': this.data, 'name':
        ↪ (!this.asc ? "" : "-") + this.data}})
      this.bgSorting = "bg-success"
    },
    deleteFromSort(){
      this.$emit('del', this.data);
      this.bgSorting = "";
    },
    filterData(data){
      if (!this.filterArr.includes(data.id)) {
        this.filterArr.push(data.id);
      } else {
        this.filterArr.splice(this.filterArr.indexOf(data.id), 1);
      }
    }
  }
}

```

```

        this.$emit("changed", {"filters": {resource: this.data, data:
        ↪   this.filterArr}})
    }
  },
  data() {
    return {
      req: '',
      ascText: "ASC",
      asc: true,
      bgSorting: "",
      filterOptions: [],
      filterArr: [],
    }
  }
}
</script>

```

Листинг 5: Column.vue

```

data() {
  return {
    className: "item d-flex flex-row align-items-center col pl-0 pr-0",
    buttonsClass: "btn-group d-flex flex-row ",
    compClass: "item d-flex flex-row align-items-center col pl-0 pr-0",
    show: false,
  }
},
methods: {
  click(event) {
    document.location = "/briefs/" + this.id;
  },
  mouseOver(event) {
    this.compClass = this.className + " bg-accent";
    this.show = true;
  },
  mouseLeave(event) {
    this.compClass = this.className;
    this.show = false;
  },
  deleteItem(e){
    e.stopPropagation();
    axios
      .delete(window.location.origin + '/api/briefs/'+this.id)
      .then(this.$emit('reload'));
  }
}

```

```
}
```

Листинг 6: TableRow.vue

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 (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).