Протокол анализа Отчета подобия Научным руководителем

Заявляю, что я ознакомился(-ась) с Полным отчетом подобия, который был сгенерирован Системой выявления и предотвращения плагиата в отношении работы:

Автор: Адамов Ильяс Ержанович

Название: Веб приложение – конкурс ППС

Координатор:Жибек Алибиева

Коэффициент подобия 1:0,4

Коэффициент подобия 2:0

Замена букв:2

Интервалы:0

Микропробелы:0

Белые знаки: 0

После анализа Отчета подобия констатирую следующее:

- ☐ обнаруженные в работе заимствования являются добросовестными и не обладают признаками плагиата. В связи с чем, признаю работу самостоятельной и допускаю ее к защите;
- ☐ обнаруженные в работе заимствования не обладают признаками плагиата, но их чрезмерное количество вызывает сомнения в отношении ценности работы по существу и отсутствием самостоятельности ее автора. В связи с чем, работа должна быть вновь отредактирована с целью ограничения заимствований;
- ☐ обнаруженные в работе заимствования являются недобросовестными и обладают признаками плагиата, или в ней содержатся преднамеренные искажения текста, указывающие на попытки сокрытия недобросовестных заимствований. В связи с чем, не допускаю работу к защите.

Обоснование:

……

…… ……

Дата *Подпись Научного руководителя*

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН
СӘТБАЕВ УНИВЕРСИТЕТІ

ОТЗЫВ

НАУЧНОГО РУКОВОДИТЕЛЯ

на дипломный проект
(наименование вида работы)
Адамов Ильяс Ержанұлы
(Ф.И.О. обучающегося)
5В070400 – Вычислительная техника и программное обеспечение
(шифр и наименование специальности)

Тема: «Веб приложение – Конкурс ППС».

Дипломная работа Адамова И.Е посвящена автоматизации конкурса ППС. Актуальность данной работы очевидна, так как проведения конкурса ППС на цифровом пространстве необходима для экономии ресурсов в Satbayev University. Данная дипломная работа представляет собой JSON API, который оказывает содействие в разработке SPA и мобильной прилжении для проведения конкурса ППС. Программа позволяет значительно сэкономить время рассмотрения кандидата на вакантную должность. Содержание работы соответствует ее названию, программа разработана согласно поставленным техническим задачам. Дипломная работа состоит из трех разделов. Первый раздел посвящен общему описанию проекта, а также техническим задачам. Во втором разделе студент описывает какие языки программирования и фреймворки существуют, тем самым оценив обьем работы и технические стороны проекта обаснованно выбрал фреймворк Laravel для решения поставленных задач. Третий раздел посвящен архитектуре проекта, где студент расширил базовую архитектуру фреймворка применив лучшие принципы и паттерны проектирования, для соблюдения чистого и понятного кода. Так же простестировал API с помощью программного обеспечение POSTMAN. Студент Адамов И.Е абсолютно готов применять свои технические навыки на производстве, и заслуживает присвоения академической степени бакалавра по специальности 5В070400 – «Вычислительная техника и программное обеспечение» и может быть допущен до защиты.

Научный руководитель
лектор
(должность, уч. степень, звание)
Ерсари І.Н
(подпись)
«__» _____2020 г.

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН

СӘТБАЕВ УНИВЕРСИТЕТІ

Институт кибернетики и информационных технологий

Кафедра "Программная инженерия"

Адамов Ильяс Ержанович

Веб приложение – конкурс ППС

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к дипломному проекту

Специальность 5В070400 – Вычислительная техника и программное обеспечение

Алматы 2020

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН

СӘТБАЕВ УНИВЕРСИТЕТІ

Институт кибернетики и информационных технологий

Кафедра "Программная инженерия"

ДОПУЩЕН К ЗАЩИТЕ
Заведующий кафедрой ПИ
канд. техн. наук, доцент,
ассоциированный–
профессор
_____Р. Юнусов
"___"_____2020 г.

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к дипломному проекту

На тему: Веб приложение — конкурс ППС

по специальности 5В070400 – Вычислительная техника и программное обеспечение

Выполнил Адамов И.Е

Научный
руководитель лектор
_____Ерсари І.Н
"_____"_____2020 г.

Алматы 2020

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН

СӘТБАЕВ УНИВЕРСИТЕТІ

Институт информационных и телекоммуникационных технологий

Кафедра "Программная инженерия"

5В070400 – Вычислительная техника и программное обеспечение

УТВЕРЖДАЮ
Заведующий кафедрой ПИ
канд. техн. наук, доцент,
ассоциированный–профессор
_____ Р.Юнусов
"_____" _____2020 г.

ЗАДАНИЕ
на выполнение дипломного проекта

Обучающемуся *Адамову Ильясу Еражанұлы*
Тема: *Веб приложение – конкурс ППС*
Утверждена *приказом проректора по академической работе № 1162 – б от "16" октября 2019 г.*
Срок сдачи законченного проекта *"_" _____ 2020 г.*
Исходные данные к дипломному проекту: *разработать JSON API для автоматизации конкурса ППС*
Перечень подлежащих разработке в дипломном проекте вопросов:
а)архитектура системы
б)паттерны проектирования
в)разработка API
г)тестирование API
Перечень графического материала (с точным указанием обязательных чертежей):
представлены 45 слайда презентации.

ГРАФИК

подготовки дипломного проекта

Наименование разделов, перечень разрабатываемых вопросов	**Сроки представления научному руководителю и консультантам**	**Примечание**
1. Анализ предметной области, разработка технического задания на проектирование архитектуры системы	20.01.2020	
2. Выбор инструмента для ведения разработки системы.	01.02.2020	
3. Разработка REST API.	10.03.2020	
4. Тестирование API	10.04.2020	
5. Написание пояснительной записки к дипломному проекту	19.05.2020	

Подписи

консультантов и нормоконтролера на законченный дипломный проект
с указанием относящихся к ним разделов проекта

Наименования разделов	**Консультанты,И.О.Ф. (уч. степень, звание)**	**Дата подписания**	**Подпись**
Программное обеспечение	Куникеев А.Д. магистр тех.наук, сениор–лектор		
Нормоконтролер	Марғұлан Қ. магистр тех.наук, сениор–лектор		

Научный руководитель_____ І.Н. Ерсари
Задание принял к исполнению обучающийся _____ И.Е. Адамов
Дата "____" _____ 2020г.

Аннотация

Данное программное обеспечение разработана для отдела Human Resources университета Satbayev University, целью которого является автоматизация и проведения конкурса ППС в цифровом пространстве. Дипломный проект состоит из трех главных разделов, а также из заключении и списка использованной литературы.

Во введении рассматривается причина автоматизации Конкурса ППС.

Первый раздел включает в себе общее описание, а также посвящен терминам и сокращениям, целям и задачам которые предстояло реализовывать.

Во втором разделе полностью описывается рабочая среда, архитектура Фреймворка и обосновывается выбранный инструмент для ведения разработки и почему выбран тот или иной подход разработки.

Третья часть проекта посвящена структуре и архитектуре проекта.

В заключении подводится итог всей работы.

АҢДАТПА

Бұл бағдарламалық жасақтама Сәтбаев Университетінің Human Resources бөліміне арналған. Негізгі мақсаты профессорлар мен оқытушылар ұйымының конкурсын сандық кеңістікте өткізу мен автоматтандыру болып табылады. Дипломдық жоба негізгі үш тараулардан, сондай–ақ қорытындыдан және пайдаланылған әдебиет тізімдерінен тұрады.

Кіріспеде профессорлар мен оқытушылар ұймының конкурсын автоматтандыру себептері қарастырылады.

Бірінші бөлім жалпы сипаттауды, сондай–ақ терминдер мен қысқартуларды, іске асырылатын мақсаттар мен міндеттерді қамтиды.

Екінші бөлімде жұмыс ортасы мен архитектурасы сипатталады және жобаны іске асыру үшін таңдалған құрал және неге таңдалғаны туралы негізделеді.

Жобаның үшінші бөлімі: бағдарламаның құрылымы мен архитектурасына арналған. Қорытындыда барлық жұмыстың қорытындысы шығарылады

Annotation

This software was developed for the Human Resources department of Satbayev University, the purpose of which is to automate and conduct a teaching staff competition in digital space. The graduation project consists of three main sections, as well as the conclusion and list of references.

The introduction discusses the reason for the automation of the faculty members competition.

The first section includes a general description, as well as the terms and abbreviations, goals and objectives that were to be implemented. The second section fully describes the working environment, the architecture of the framework and justifies the chosen tool for developing and why this or that development approach is chosen.

The third part of the project is devoted to the structure and architecture of the project.

In conclusion, the result of all the work is summarized.

Содержание

Введение

Университет Satbayev University ежегодно обьявляет конкурс на замещение некоторых вакантных должностей. В конкурсе могут участвовать:

- Заведующие кафедрами.
- Профессора
- Ассоциированные профессора
- Ассистент – профессора
- Сениор–лекторы
- Тьюторы
- Ассистенты
- Лектора

Соискатели могут предоставлять резюме, документы касающиеся образования, сертификаты, рекомендации от действующих сотрудников, документы подтверждающие профессиональный уровень кандидата, сертификаты международного стандарта, копию диплома, а также множество других грамот. В целом существует точный список документов для участия в конкурсе.

Проведения конкурса имеет в плане ресурсов неприятную сторону. А именно каждый год университет из своего бюджета тратить неимоверное количество бумаг на распечатывания резюме членам коммиссии. Представьте сколько денежных затрат составит для распечатывания резюме на каждого члена комиссии. А также сотрудники отдела кадров тратят свою энергию на множество проверок, подготовок, фильтрации документов.

Соответственно данный дипломный проект демонстрирует результат разработки JSON API который автоматизирует логику серверной части, предназначенный для проведения конкурса ППС.

1 Исследовательский раздел

1.1 Цель разработки системы

Главная цель дипломного проекта является упрощение процесса рассмотрения соискателей на вакантную должность, путем голосования конкурсной комисии.

Проект должен избавить от ненужной печати множества экземпляров резюме для членов комиссии. Члены комиссии в будущем должны быть оснащены планшетами на заседании, через которые им будут доступны все анкетные данные кандидата.

А также система должна автоматизировать ряд других обязанностей, которые лежат на плечах сотрудников университета.

1.2 Определения термин и сокращении

Сокращение или термин	Определение
PHP	Hypertext Preprocessor
API	Application Programming Interface
JSON	JavaScript Object Notation
HTTP	HyperText Transfer Protocol
SQL	Structured Query Language
PSR	PHP Standard Recommendation
CRM	Customer Relationship Management
ERP	Enterprise Resource Planning
SOLID	Принцип проектирования
GRASP	Шаблон проектирования
Gof Pattern	Шаблоны проектирования “Банда четырех”
Composer	Пакетный менеджер зависимостей языка программирования PHP

1.3 Технические задачи

Инструмент должен иметь внутреннюю систему аутентификации и авторизации с определением ролей: администратор, сотрудник отдела HR, заведующий кафедры, председатель, проректор, ректор, директор института.

Автоматизация задач администратора:

Модуль администратора должен состоят из нескольких подмодулей, такие– как: Пользователи, Кандидаты, Подразделения, Собеседования, Вакантные позиции. Каждый из них должен решать определённые задачи в своем контексте.

Ресурс Пользователи:

— Создать пользователя.
— Указать ФИО, телефон, адрес, ИИН, email.
— Загрузить фото.
— Указать роль в системе.
— Указать подразделения.
— Указать должность пользователя.
— Удалить пользователя.
— Удалить из хранилища все загруженные документы, а также фотографию.
— Редактировать пользователя.
— Получить список всех пользователей.
— Получить информацию определенного пользователя.

Ресурс Кандидаты:

— Получить список всех кандидатов.
— Получить определеннего кандидата из базы по id.
— Удалить кандидата из базы.

Ресурс Подразделения:

— Создать подразделения.
— Редактировать подразделения.
— Удалить подразделения.
— Получить список всех подразделении.
— Получить информацию по id подразделении.

Ресурс Собеседования:

— Создать собеседование.
— Отменить собеседование.
— Удалить собеседование.
— Получить список всех собеседовании.
— Получить данные по id собеседовании.
— Завершить собеседования.
— Редактировать собеседования.

Ресурс Вакантные позиции:

— Создать позицию.
— Редактировать позицию.
— Удалить позицию.
— Получить список позиции.
— Получить данные по id позиции.

Ресурс График:

— Получить список всех институт и кафедр.

— Выбрать время проведения конкурса для выбранного института.

Кандидат. В первую очередь кандидат должен загрузить все необходимые документы, также указать свои данные, а именно:

ФИО:

— Поле “Фамилия” и “Имя” должны быть указаны в обязательном порядке.

— Максимальная длина полей “Фамилия” и “Имя” должны составлять 25 символов, поле “Отчество” является опциональным.

Номер телефона:

— Поле “Телефон” обязателен к заполнению.

— Максимальная и минимальная длина номера телефона должна составлять 11 цифр

E–mail:

— Поле “E–mail” должен быть типом email.

— Поле “E–mail” обязателен к заполнению.

ИИН:

— Поле “ИИН” обязателен к заполнению.

— ИИН должен быть уникальным, если система будет возвращать сообщения “в базе существует указанный ИИН”, но пользователь уверен в правильности заполнения ИИН, вернуть ошибку чтобы кандидат связался с администрацией сервиса или же по указанному номеру телефона отдела HR.

— Минимальная и максимальная длина ИИН должна составлять 11 цифр

Фотография:

— Фотография кандидата должна быть загружена.

— Кандидат должен загрузить свою фотографию в формате: png, jpg, иначе вернуть ошибку о неправильном формате.

— Максимальный обьем памяти фотографии должен составлять 10мб

Департамент:

— Кандидат обязательно должен указать депратамент.

— Указанный департамент должен существовать в базе, если кандидат каким–то образом указал несуществующий департамент (маловероятно) вернуть ошибку.

Вакантная позиция:

— Кандидат обязательно должен указать вакантную позицию.

— Вакантная позиция должна существовать в базе, если кандидат каким-то образом указал несуществующую позицию (маловероятно) вернуть ошибку.

Документы:

— Все документы кандидата должны быть загружены в формате: doc, docx, ppt, pptx, pdf.

— Максимальный объём памяти каждого документа должен составлять 15мб.

Автоматизация задач сотрудников отдела Human Resources:

Сотрудниками отдела Human Resources должны проводится всесторонние проверки загруженных документов кандидата. Если с документами кандидата все в порядке, HR–у необходимо:

— назначить собеседования.

— уведомить кандидата по email о назначенной первой собеседовании, который соответственно будет проводиться отделом Human Resources.

— загрузить заключение, а также результаты тестирования.

— необходимо обновить статус заявки и передать данные кандидата, соответствующему департаменту, чтобы руководитель департамента мог со своей стороны дать оценку качествам кандидата.

Как только заведующий кафедры даст свое заключения о кандидате, HR–у следует:

— назначить завершающее собеседование.

— составить список членов коммисии.

— уведомить членов коммисии и соискателя по email о предстоящей собеседовании.

Автоматизация задач заведующих кафедр и директоров институтов:

Чтобы заведующие кафедр и директора институтов могли ознакомиться с резюме кандидата и соответственно оценить профессиональные качества кандидата, а также со своих сторон дать заключения о кандидате.

— Получить список нужных кандидатов.

— Назначить второе собеседование кандидату.

После дать оценку:

— о профессиональных компетенциях

— указать дисциплины соискателя

— оценить научные достижения, дикцию, поведения и так далее.

Автоматизация задач Председателя:

— Перед председателем должен открываться список соискателей, согласно графику приема Институтов/Кафедр (График составляется сотрудниками HR).

— При нажатии на соискателя председателем – у всех членов комиссии автоматический должны открываться анкета по соискателю.

— Решением председателя (в случае болезни члена комиссии) нажимает кнопку закончить.

— Голосование считается законченным при наличии всех голосов членов комиссии.

— В конце рассмотрения соискателей по кафедре система выдает отчет для распечатки и подписи членами комиссии.

Автоматизация задач участников комиссии:

— Как только председатель комиссии активирует назначенное собеседование, все участники могут получить доступ к информациям кандидата.

После всей сторонии иследовании кандидата участниками комиисии, соответственно они должны голосовать, то есть нужно:

- Реализовать функционал кнопки «Против».
- Реализовать функционал кнопки «За».

2 Технологический раздел

2.1 MySQL и XAMPP

SQL это стандарт для управления данными. Данные в свою очередь хранятся в реляционных базах данных. Существуют несколько реализации SQL смотреть рисунок 1, самые популярные и распространённые это: Oracle, Microsoft SQL Server, PostgreSQL и другие. Несмотря на то, что реализации SQL много, они все между собой похожи. Зная одну реализацию SQL, можно без каких–либо трудностей адаптироваться в другой реализации.

Рисунок 1 – реализации SQL

В дипломном проекте используется MySQL смотреть рисунок 1.1 вместе со сборкой XAMPP. XAMPP в данной системе выступает в качестве локального сервера и будет эмулировать работу хостинга. В сборку XAMPP входит веб–сервер Apache, MySQL, так же серверный язык PHP. MySQL позволяет хранить информацию структиурирванно. Имеется бесплатная и платная версия данной СУБД. Но больше всего используется в серверных сборках XAMPP, LAMP, MAMP.

Рисунок 1.1 – MySQL

2.2 Языки программирования

Для начала необходимо определить на каком языке программирования будут решаться предстоящие задачи.

PHP – популярный и массовый, на котором создано большое количество систем. Изначально PHP был разработан для решения простых задач. Со временем стал набирать большую популярность и пользоваться спросом среди разработчиков, которые использовали ранее другие ЯП.

PHP оброс пакетным менеджером Composer, пространством имен, стандартами и дополнительными библиотеками в большом количестве [5]. Также стал профессиональным инструментом, на котором можно писать не только гостевые книги, но и полноценные проекты и большие порталы.

В дальнейшем на нем разработали множество CMS и фреймворки, также появилась собственная типизация, которая сильно отличается от абсолютной строгой типизации.

Помимо PHP выбирают и другие языки программирования Ruby, Python, NodeJS, Go и другие.

Ruby был популярен в 2013, 2014 годах. Со временем эпоха Ruby On Rails начала проходить, когда на Хабре перестали публиковать статьи про него и в дальнейшем Ruby пропал с поля зрения. Но это вовсе не означает, что на Ruby нельзя разрабатывать хорошие проекты.

Большая часть программистов на Python являются разработчики моделей машинного обучения и нейронных сетей. Для Python написано большое количество математических и статистических библиотек, которые позволяют писать персептроны и все что связано со статистикой, машинным обучением и распознованием лиц.

Несмотря на что Python можно использовать во многих сферах программирования также имеет ряд недостатков таких как: низкая скорость, динамическая типизация.

Java является лучшим решением для Enterprise разработки с использованием обьектно–ориентированного подхода, также имеет строго типзированную систему и не позволяет писать процедуру или функцию. В итоге он дает очень хорошую систему типизации которая подойдет многим. Благодаря Java в версиях PHP 7.x и выше появились типизация и генераторы.

На NodeJS можно разрабатывать микросервисы, веб–сокеты, демонические системы. Демонический это значит – демоны которые один раз запускаются, в итоге они так и запущенными остаются, где в дальнейшем принимают соединения и поддерживают их. NodeJS не требует компиляции как и сам JavaScript. Однако существует проблема, на которую в последние годы ругались многие разработчики. Проблема заключается в утечки памяти, раз в сутки требуется перезапускать скрипты, так как NodeJS потребляет все больше и больше памяти [8].

Go появился в 2009 году и обрел популярность на просторах интернета благодаря маркетингу, является низкоуровневым, примитивным, компилируемым ЯП, очень хорошо работает с потоками и держить нагрузку, порой работает напрямую с байтами [7]. Отличительная черта в том, что он содержит около двадцати синтаксисчеких конструкции, что позволяет программировать сразу после прочтении книги или же документаци. В нем отсутствуют exceptions которые многие любят в других языках, но больше всех поразил особой системой “goroutine” которая позволяет очень удобно писать многопоточный код. Еще один огромный плюс Go в том, что на нем можно написать код, после скомпилировать из него .exe или бинарник затем загрузить на сервер и программа будет работать без сбоев.

Каждый язык программирования имеет ряд преимущств и недостатков, но каждый из них заточен решать определенные задачи в своем контексте. Для данного проекта был выбран язык программирования PHP, так–как отлично подходить для решения предстоящих задач. Для ускорения процесса разработки будет применяться Фреймворк.

2.3 Фреймворк.

Фреймворк – это огромная структура, заранее заготовленных классов и методов, а также сложных программных решении, которые будут являтся ядром проекта, их много и каждый из них интересен. Отличительной чертой от библиотеки является комплексность решения задач. С технической точки зрения это механизм, позволяющий автоматизировать создания какой–либо системы и писать частные случаи кода. Философия любой технологии необходима программисту для понимания дальнейшего взаимодействия с ним и области его применения. На форумах разработчики часто пишут “Выбирайте инструмент под

задачу” так–как выбор правильной платформы под текущую задачу очень важна. Но ответов как правильно выбирать инсртумент для решения задач в Интернете очень мало. Одними из таких крупных систем являются Laravel и Symfony смотреть рисунок 2 [6].

Рисунок 2 – Фреймворк Symfony и Laravel

Существуют также средние, но быстрые платформы как YII Framework смотреть рисунок 2.2 [2].

Рисунок 2.2 – Фреймворк YI

Есть микроплатформы, такие как: Slim, Phalcon, Lumen смотреть рисунок 3 [9].

Рисунок 3 –Микроплатформы Slim, Phalcon и Lumen

Существенная разница между ними состоит в построении архитектуры и что они предоставляют собой.

2.4 Как выбрать платформу?

Существуют критерии по выбору платформы для решения задач, а именно: монолитность, компонентность, совместимость, внутренняя архитектура, расширяемость, поддержка, отношения авторов, паттерны и принципы проектирования. Если инструмент монолитен, в какой–то момент вопросы по архитектуре могут привести к затруднениям. Это все равно что писать проект на WordPress и вдруг захочется перейти с него на какой–нибудь другую платформу, и тут же становится ясно, что с WordPress ничего не переносится. То что в нем несколько тысячь плагинов, никакого результата не дадут. Все это приводит к тому что надо переписать проект заново, вместо монолитного лучше сразу использовать компонентый фреймворк, когда активно используется Composer.

Стоит так же проверить, на совместимость со стандартами. В мире PHP приняты некие PSR стандраты, например PSR 7 для request/response, PSR 15 для контроллеров и middleware, PSR 16 для кэша. Совместимость дает гибкость, при несовместимости придется для каждого стандартного PSR интерфейса писать собственный костыль чтобы сконфигурировать структуру.

Еще интересно как с внутренной архитектурой. Когда человек пишет свой код на фреймворке, он может проконтролировать процесс разработки. Если утруждаться над архитектурой, надо искать такой инструмент, который соответственно в архитектурном плане лучше сделан. Если у системы все хорошо с принципами и паттернами, в этом случае дозволено расширять и дописывать к нему своим любые компоненты.

Даже если расширяемый, возникает вопрос “Обновляется ли составная часть?”. Бывали случаи что используемый компонент описан одним человеком и разработчик уже 3 года не заходил на GitHub. Компонент такого рода очень опасен и рискованно внедрять в свой проект, поэтому стоит быть внимательным. YII 2.0 был спрограммирован в 2014 году, а YII 3 еще не вышел. В текущей версии архитектурных сильных изменении нету, все в пределах обратной совместимости и фикса багов, несколько переменных переименовали и пару интерфейсов добавили. Если новая версия выйдет можно подумать, если нет, то какой смысл делать проект на монолитном YII 2.0 шести летней давности, при этом жить в страхе что вот–вот скоро выйдет 3 версия, и придется проект переписывать заново.

Авторы разрабатывают свои инструменты либо в свободное от работы время как делается YII 2.0, либо фултайм как Taylor Otwell. Taylor пишет Laravel так–как это у него основная работа и старается улучшать свой продукт. Каждый месяц выпускает версию 5.x, 6.x, 7.x, так же и автор Symfony, Fabien Potencier тоже старается выпускать версии 3, 4, 5.0

Лучше выбирать инструмент, к которому относятся очень серьезно. Есть два фундаментально противоположных подхода к разработке программы. Первый подход называется Rapid Application Development – это быстрая разработка софта. Такой подход приводит к бездумной и неконтролируемой разработке, программист легко заблудится между файлами. К примеру, если вы решили написать большой портал и за этим строго не следить, получится много мегабайтный хаос, поддерживать такие проекты не будут, потому что структура очень плохо организована, чем в той же WordPress, и подход разработки намного хуже. RAD подход подойдет для прототипов.

С прототипами есть такая вещь что настоящий прототип не страшно писать любым кодом, при условии что прототип программист перепишет соблюдая все паттерны и принципы. Когда проект крупнеет, только в этом случае возникают достоинства и недостатки разных подходов. До тех пор пока программист не запишет консольные команды для бизнес логики, он начинает думать как отрефакторить, чтобы логика работала универсально.

Откуда взялись паттерны и принципы SOLID, GRASP, GoF? Разработчики, создавая проект обратили внимания на высокую загруженность программы, после внедрения различных подходов решили писать слабосвязанный код отказавшись от хаотичных связей. Вывели такие принципы, что код нужно разделять по ответственностям, минимизровать число связей, разделять по слоям, и избегать циклических связей.

Многие программисты сходятся на мнении что фреймворк Symfony работает медленно. Подобное мнение останавливает разработчиков от использования Symfony. Например Вконтакте и Facebook являются HighLoad–ными системами, поэтому инженеры данных IT–гигантов разработали собственную версию PHP, который функционирует без обьектно–ориентированного подхода и гаранитирует быстродействие. Проекты средней сложности не требуют быстродействии. В плане быстройдействии YII Framework хороший выбор, Symfony больше подходит для корпоративных проектов.

Нужно понимать что архитектура придумана, чтобы упрощать, но не наоборот усложнять. Если проект очень простой, где требования руководство гласить сделать все побыстрее, например блог, то надо брать WordPress. Если требуется все сделать быстро, но на фреймворке, где все идет с коробки в этом случае стоит брать YII Framework. Если никто никуда не спешить и никаких авралов нет, есть ресурсы и время подумать, а также нужно программировать по строгости и основательно, для таких задач следует брать Enterprise инструмент по серьзнее.

Крупные проекты в лучших IT компаниях пишут на Java с его Spring или же C# .NET, они являются стандартами для Enterprise разработки. Все банковские системы написаны на таких крупных типизированных языках, где присутствуют готовые библиотеки и хорошая архитектура, которые в свою очередь позволяют

разработчикам обращаться с любым проектом на “ТЫ”. Если не хочется брать Java или C#, в мире PHP в отличии от общеупотребительных инструментов есть только Zend и Symfony которые рассчитаны под архитектуру сложных проектов.

Тем не менее есть универсальный лучший инструмент где можно писать и плохо, и хорошо в плане архитектуры, это Laravel, он звено между YII и Symfony. В нем дозволено писать код как в YII с фасадами, одновременно можно писать код Enterprise уровня, имеется очень хороший контейнер, который поддерживает хранения сервисов. Он популярный, универсальный, совместимый, расширяемый и внутренности написаны хорошой архитектурой, в этом плане золотая середина.

Проанализировав все инструменты для ускорения процесса разработки, для дипломного проекта выбран Laravel Framework[1].

2.5 Laravel и экосистема

Laravel один из популярных PHP фреймворков, но представляет прежде всего экосистему. Например, YII 2.0 – это обыкновенный инструмент и больше к нему никаких сервисов нет. Laravel – это не одна папка со всеми файлами. Экосистема данного инструмента состоит из трех основных вещей, это сама документация, репозиторий который содержит исходный код шаблона и репозиторий Illuminate, который содержит такие компоненты как: сессия, роутинг, redis, очередь, pipline, пагинация, нотификация, mail, log, http, hashing, events, encryption, database, то есть по анологии экосистемы Symfony.

Также у него имеются свои дополнительные сервисы такие как: Vapor, Forge, Envoyer, Horizon, Nova, Echo, Lumen, Homestead, Spark, Valet, Mix, Cashier, Dusk, Passport, Scout, Socialite, Telescope, Tinker. Таких сервисов другие платформы не имеют, но при этом у этого есть некая дополнительная черта, некоторые из этих сервисов у Laravel порой бывают платными и это вполне аргументированный ход со стороны разработчиков данного программного продукта, потому что эти сервисы работают на настоящих и очень дорогих серверах, поэтому требует поддержки и обновлении. Если пользоваться системой веб–сокетов , придется отчислять немного денег для того чтобы все работало.

При переходе на репозиторий на GitHub по адресу https://github.com/laravel/laravel можно увидеть лишь поверхность айсберга этой экосистемы, то есть стандратный шаблон под проект[3]. Это не сам фреймворк. На самом деле он находится в элементе framework по пути https://github.com/laravel/framework дальше в папке src/illuminate есть очень много подпапок. Собственно эти подпаки являются ссылкой на другие репозитории, то есть в GitHub все они находятся в одной папке src, но на самом деле все они

находятся в репозитории, который называется Illuminate. Каждая папка из папки src это определнный компонент который расположен отдельными названиями: view, validation, translation и так далее. Все компоненты находятся отдельными репозиториями, но при этом подключены к полному репозиторию.

Все эти выше перечисленные компоненты являются самодостаточными, они спрограммированы таким образом, что их можно подключать отдельно к любому другому фреймворку. Компоненты написаны по принципу SOLID, все зависимости работают через интерфейсы, настройки можно конфигурировать у самих же этих компонентов через контейнер внедрения зависимостей. Соответсвенно имеется собственный контейнер, присутствуют системы для работы с консолью, шинами, аутентицикацией. Тот же самый Zend Framework используют точно такие же подрепозитории и у каждого компонента есть своя документация[4]. Это очень хорошо, удобно и дает много пользы PHP сообществу которое это все использует.

То есть смотря на все такие фреймворки как: Zend, Laravel, Symfony можно понять, что все они построены по компонентной структуре и представляют собой набор отдельных библотек, живущих в отдельных репозиториях. И все компоненты можно использовать без установки самого фреймворка, просто подключив с помощью Composer.

Когда появился Composer, тренд у PHP разработчиков пошел на использования компонентых систем. Все фреймворки которые работают по компонентной системе обретают популярность, другие монолитные наоборот теряют популярность. Автор фреймворка Taylor Otwell. Кто–то может с ним поспорить что он особо налегает на трейты, на магические методы в своей системе, но программирует он профессионально. Если сравнивать с другими фреймворками, на котором программируют в основном делитанты, потому что смотришь на них и не понимаешь, как такое можно было написать в здравом уме. Но многие они были написаны в те времена когда не было еще такие понятие как: нормальный код, рефакторинг, Composer, поэтому они не используются и нишу заняли другие фреймворки.

Если открыть документацию, то в первой же странции написано что фреймворк для веб–ремесленников. Ремесленник человек который разбирается в своей сфере и любить свою работу. Важно следующее: любой инструмент который мы будем использовать, чтобы он нам помогал, но никак не мешал. Laravel Framework представляет из себя золотую середину по сложности, по порогу вхождения, и экосистеме. Данный инструмент популярен так как его автор всегда Taylor Otwell прислушивается комьюнити. Соответсвенно, чтобы написать такую программную платформу которые полюбили бы многие другие программисты, надо знать чего они хотят.

Перед ним стояла задача провести опрос, проанализировать PHP сообщество и выяснить каким образом он может помочь и порадовать. Оказалось что людям

нравится инструмент с хорошой архитектурой, который соответствует стандартам, принципам , тому же SOLID–у и GRASP–у. То есть чтобы сделать глобальный инструмент, нужно очень хорошо разбираться в теме спроса и понимать чего хотят от вас люди. Taylor Otwell оказался в теме этого спроса, и написал инструмент. Очередной причиной популярности данного инструмента – это большой маркетинг.

3 Проектная часть

3.1 Расширения базовой архитектуры и паттерны проектирвования

В данном проекте по умолчанию структура плоская, исходный код складывается в корень фреймворка в такие базовые директории как: Resources, Requests, HTTP, Controllers, Exception, Mail, Providers, Console, Entity, Middleware и так далее. Принято решения расширить базовую иерархию данного программного обеспечение также использовать промежуточную архитектуру где выделяются сервисы и репозитории.

Система используется в качестве JSON API который лишь передает и обрабатывает данные. В системе местами повторяются определенные бизнес–логики, которые соответственно не должны повторятся по принцпу DRY, следовательно чтобы избежать плохого тона, в проект внедрены программно–архитектурные сервисные слои.

Существуют несколько способов использования сервисов, а именно статический сервис Helper, Сервисный обьект, Внедрение зависимостей.

В первую очередь необходимо отделить файлы проверки в класс валидации так–как не входит в обязанность данного слоя, но тем не менее фреймворк предоставляет отличную возможность, и все проверки проходят до активизации контроллеров и позволяет передать лишь отфильтрованные данные.

Второй шаг заключается в создании папки Services, в которой будут находится все нужные сервисы. Способ через внедрение зависимостей подходит для решения поставленных задач. Данный способ экономит время, так–как сервис инжектируется через конструктор контроллера, требуется лишь указать тип.. Данное решение даст чистоту и читабельность контроллеров, и вся логика сосредаточена в сервисах, в противном случае контроллеры станут нечитаемыми. На прострах интернета во многих источниках можно встретить неправильное представление о паттерне “Репозитории” или “Хранилище” смотреть рисунок 3, якобы данный паттерн предназначен для обновления и создания записи. На самом деле данный паттерн приследует другие цели. Роль паттерна Репозитори заключается лишь в получении некоторых данных из базы. Если контроллер

обращается к разным сервисам чтобы упрощать и делать контроллер очень читабельным и тонким, соответственно сервис в свою очередь должен обращаться к репозитории лишь для извлечения данных.

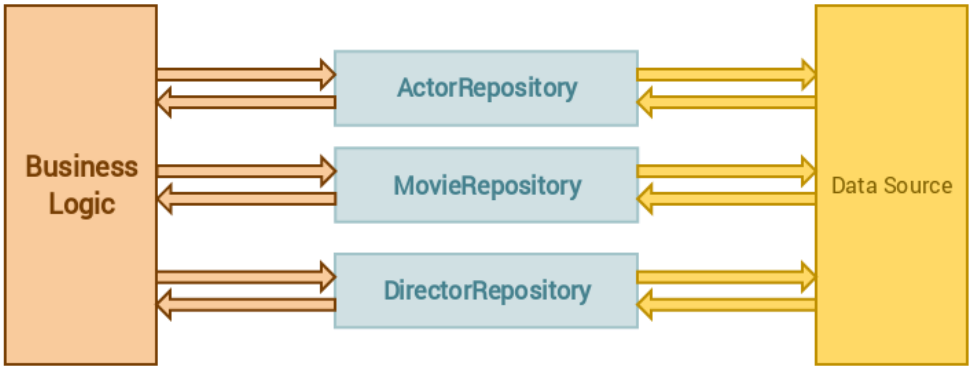

Рисунок 3.1 – Паттерн Repository

Для внедрения паттерна “Репозиторий” в проекте проделаны очень простые шаги. Во первых организиована папка Repositories, данная папка содержит интерфейсы и сами файлы хранилищ определенных сущностей.

Во вторых внутри папки Interfaces созданы интерфейсы, которые необходимо реализовать при имплементации интерфейсов. В третьих созданы классы которые реализуют необходимые интерфейсы, в самих классах необходимо придерживаться имплементации. Чтобы использовать все репозитории, необходимо их внедрить в контроллеры. Данную задачу следует положить на плечи сервис–контейнера. Для этого с помощью команды создания провайдера смотреть рисунок 3.2.

```
C:\xampp\htdocs\satbayevhunt>php artisan make:provider DepartmentRepository
```

Рисунок 3.2 – Команда для создания сервис провайдера

Создается отдельный провайдер чтобы соблюдать чистоту кода, после в методе register регистрируются все интерфейсы и классы.

Одной из главных целей данного проекта является построения API который позволит передавать данные разным клиентам. С релизом пятой версии фреймворка, разработчики получили очень много интересных возможностей. Taylor Otwell среди этих возможностей представил инструмент, так называемый API ресурс который заменил Fractal. Задача API ресурса заключается в преобразовании коллекции в удобный JSON ответ. Такую задачу решают Serializer и Fractal, но API ресурс среди других решении является самым удобным и лучшим

для преобразования json ответов. Для генерации ресурса необходимо вести команду генерации ресурса смотреть рисунок 4:

```
php artisan make:resource User
```

Рисунок 4 – Команда генерации json ресурса.

После введения команды в директории Resources сгенерируется ресурс. Например ресурс UserListResource данный файл содержит функцию toArray() , цель которой вернуть массив атрибутов. Достаточно из сервиса или контроллера вернуть коллекцию или данные одного обьекта. Если протестировать с помощью POSTMAN клиент получить нужные данные для отображения в представлении, следует лишь отправлять правильные запросы.

3.2 REST API

API это некоторый программый интерфейс. Данное понятие находится не только в контексте веб разработки но и распространяется на любую разработку.

API присутствуют практический у любых программных элементов, такие как наушники, телефоны, микропроцессоры и так далее. Коммуникация между различными системами происходит посредством API. Данное понятие более глобальное и существуют большое количество систем которые зачастую обмениваются данными через API. Относительно–популярный случай API в контексте веб–программирования является REST API.

Representational State Transfer впервые данная абриевиатура была введена Роем Филдингом в 2000 году в его десертации, он предложил возможность коммуницировать разные части сервисов с помощью HTTP запросов. REST API на данный момент является одним из самых популярных архитектурных надстроек HTTP протокола.

Также имеет определенные принципы, первым из которых является клиент–серверная архитектура, второй принцип гласит о том что сервер не должен хранить состояние клиента. Второй принцип сделан для того чтобы расширить возможности сервера, таким образом можно обойти ограничения по памяти хранения данных. REST API активно использует request методы GET, POST, PUT, DELETE протокола HTTP и его статусы. Каждый из этих методов занимается своей задачей GET–получает, POST–записывает, PUT–обновляет, DELETE–удаляет данные.

Немало важную роль играет URL. URL тоже является основой REST API. При помощи URL сервер понимает с какими обьектами работать. При обмене данными

по REST API используются формат данных JSON который позволяет, клиентам на Android или iOS работать с данными напрямую. JSON эта особая нотация для передачи данных клиенту. При работе по протоколу REST, PHP не возвращает обычный массив, вместо этого используются функции json_decode и json_encode. Laravel предоставляет возможность вернуть JSON ответ из любого контроллера. Однако при использовании файла web.php который предназначен для регистрации маршрутов классического подхода разработки сайта с шабоном MVC могут возникнуть сложности другого характера. POST запросы будут валидировать XSRF токены или будет мешать работа сессии.

Разработка REST API гласит чтобы система работала без сессии, cookie, флэш сообщении, и других элементов которые используются в классическом варианте работы сайта. Соответственно должна быть отдельная аутентификация, которая будет работать по токенам. При регистрации пользователя генерируется токен подтверждения email. Помимо токена для подтверждения email существует токен для аутентификации, который позволяет извлечь данные человека из базы. Токены следует хранить таким образом, что при каждой смене пароля, или же при истеченнии времени не исчезали.

В Laravel есть отдельный файл конфигурации, который назван api.php. Настройки web.php и api.php содержатся в файле провайдеров а именно в директории RouteServiceProvider. RouteServiceProvider при загрузке нашей системы парсить все маршруты. В первую очередь парсятся маршруты для API , после маршруты для Web. Web роуты те маршруты которые перечислены в routes/web.php под namespace–ом Controllers.

Маршруты для API регистртируются под префиксом api, и будут автоматический работать с адресов, которые начинаются с подстроки api. Маршруты файла web.php работают с посредником web, маршруты файла api.php работают с посредником api. В файле Kernel.php зарегистрированы несколько групп посредников для обработки разных запросов. Соответственно перечислена группа web которая позволяет кодировать и декодировать cookie, аутентифицировать, перебрасывать ошибки, верифицировать XSRF токен через сессию.

Данная группа работает, когда на сайт обращаются через методы контроллеров которые перечислены в файле маршрутов под названием web.php. Помимо группы web, есть и другая группа который называется api.

Группа api не содержит конфигурации как группа web, но присутствует фильтр который ограничивает число запросов с одного ip адреса.Все правила маршрутизации, которые относятся API следует помещать в файл api.php.

3.3 Тестирования API с помощью POSTMAN

Цель программного обеспечение Postman, помочь программисту протестировать API. Данный инструмент позволяет без вмешательств клиентной стороны протестировать HTTP запросы смотреть рисунок 4.1.

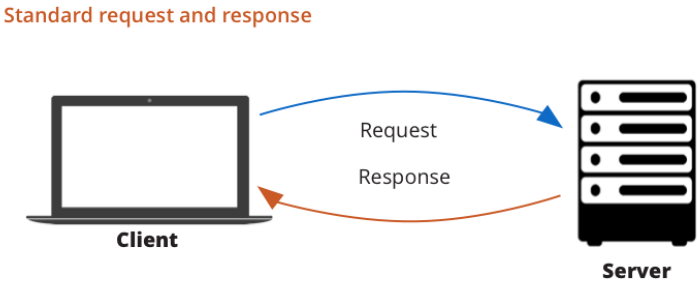

Рисунок 4.1 – HTTP запросы

Таким образом все маршруты которые прописаны в файле api.php были протестированы данным инструментом. Например запрос GET по адресу http://localhost:8000/api/admin/users возвращает данные пользователей смотерть рисунок 5, запрос попадает в контроллер где в дальнейшем обрабатывает и возвращает коллекцию пользователей.

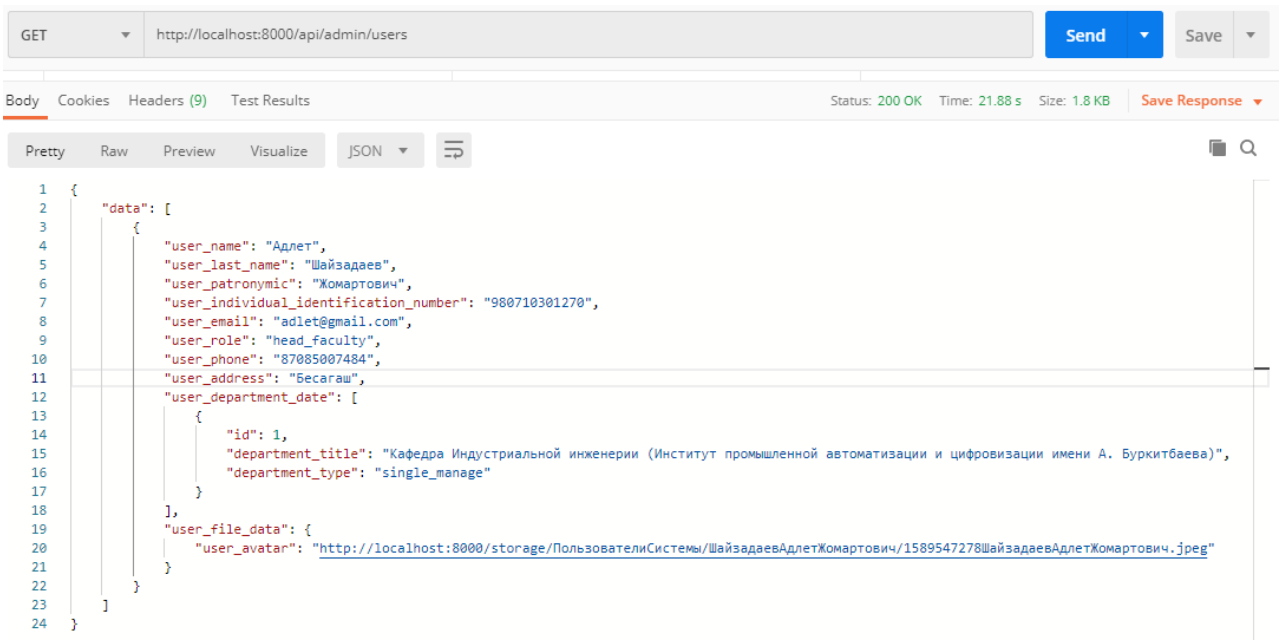

Рисунок 5 – Список пользователей

Заключения

В рамках данной дипломной работы был построен JSON API который предназначен для автоматизаци конкурса ППС для отдела Human Resources. Цель программы:

1)автоматизировать подачу заявки на вакантные места

2)снизить затраты университета Satbayev University.

3)снизить обьем работы сотрудникам.

Для построения API был выбран язык PHP а также один из лучших его компонентных фреймворков Laravel, в качестве СУБД MySQL, и веб–сервер Apache. Для соблюдения чистоты кода и оптимизации данной системы были внедрены лучшие паттерны проектирования. Также данный JSON API протестирован инструментом Postman.

Список использованной литературы

1. Документация фреймворка Laravel – https://laravel.com/docs/7.x
2. Документация фреймворка YII – https://www.yiiframework.com/
3. Репозитории Laravel – https://github.com/laravel/laravel
4. Документация фреймворка Zend – https://framework.zend.com/
5. Документация Composer – https://getcomposer.org/doc/00-intro.md
6. Документация Symfony – https://symfony.com/
7. Документация Go – https://golang.org/
8. NodeJS утечка памяти – https://habr.com/ru/company/nordavind/blog/195494/
9. Документация фреймворка Slim – http://www.slimframework.com/

Приложение А

Текст программы

1. *Interview.php*

```
<?php

namespace App\Entity\Interview;

use App\Entity\Candidate\Candidate;
use Carbon\Carbon;
use Illuminate\Database\Eloquent\Model;
use Jenssegers\Date\Date;

class Interview extends Model
{
    protected $table = 'interviews';

     protected $fillable = ['id', 'level', 'building', 'cabinet', 'interview_start', 'status', 'candidate_id', 'user_id'];

    public function members()
    {
        return $this–>hasMany(Member::class,'interview_id','id');
    }

    public function candidate()
    {
        return $this–>hasOne(Candidate::class,'id','candidate_id');
    }

    public const STATUS_FINISHED = 'finished';
    public const STATUS_WAIT = 'wait';
    public const STATUS_IN_PROCESS = 'in_process';
    public const STATUS_CANCELED = 'canceled';

    public const LEVEL_ONE = 'hr_interview';
    public const LEVEL_TWO = 'head_faculty_interview';
    public const LEVEL_THREE = 'final_interview';

    public static function createInterviewByHR($building, $cabinet, $candidateId, $authorId)
```

```
        {
        return static::create([
            'level' => self::LEVEL_ONE,
            'building' => $building,
            'cabinet' => $cabinet,
            'status' => self::STATUS_WAIT,
            'candidate_id' => $candidateId,
            'user_id' => $authorId,
        ]);
    }

    public function setDataAttribute($value)
    {
        $date = Carbon::createFromFormat('y–m–d H:i:s', $value)–>format('Y–m–d H:i:s');
        $this–>attributes['interview_start'] = $date;
        $this–>save();
    }

    public function getNormalFormatDateTime()
    {
      return Date::parse($this–>interview_start)–>format('j F Y г. H:i');
    }
}
```

2. *InterviewService.php*

```
<?php

namespace App\Services\Interview;

use App\Entity\Interview\Interview;
use App\Http\Requests\Hr\CreateInterviewByHrRequest;
use App\Repositories\Interfaces\InterviewRepositoryInterface;

class InterviewService
{
    private $interviewRepository;
    public function __construct(InterviewRepositoryInterface $interviewRepository)
    {
        $this–>interviewRepository = $interviewRepository;}
    }
```

```
    public function createInterviewByHr(CreateInterviewByHrRequest $request)
    {
        $interview = Interview::createInterviewByHR(
            $request['building'],
            $request['cabinet'],
            $request['candidate_id'],
            $request['user_id']
        );
        $interview–>setDataAttribute($request['interview_start']);
        return [
            'building' => $interview–>building,
            'cabinet' => $interview–>cabinet,
            'interview_start' => $interview–>getNormalFormatDateTime(),
            'candidate_name' => $request['candidate_name'],
            'candidate_email' => $request['candidate_email']
        ];
    }

    public function getAllInterviewsForHr()
    {
        return $this–>interviewRepository–>getHrInterviews();
    }

    public function getInterviewByIdForHr($id)
    {
        return $this–>interviewRepository–>getHrInterview($id);
    }

    public function deleteInterview($id)
    {
        $interview = $this–>interviewRepository–>getHrInterview($id);
        $interview–>delete();
    }

}
```