Министерство образования Республики Беларусь

Учреждение образования

«Могилевский государственный университет продовольствия»

Кафедра автоматизации технологических процессов и производств

**ИООО «ЭПАМ Системз»**

**Отчет по преддипломной практике**

Специальность 1 – 40 05 01 Информационные системы и технологии (по направлениям)

Направление специальности 1 – 40 05 01 – 11 Информационные системы и технологии (в пищевой промышленности)

|  |  |  |
| --- | --- | --- |
| Руководитель практики  от УО «МГУП», доцент, к.т.н | \_\_\_\_\_\_\_\_\_\_\_  Личная подпись  Дата | И.В. Акиншева |
| Руководитель практики от  ОАО «Савушкин продукт»,  Начальник отдела программирования | \_\_\_\_\_\_\_\_\_\_\_  Личная подпись  Дата | М.Г. Концевой |
| Выполнил  студент группы ИСиТ-151 | \_\_\_\_\_\_\_\_\_\_\_  Личная подпись  Дата | В.А. Пантюхов |

Могилев 2019

**Содержание**

Введение 3

1 Индивидуальное задание 4

1.1Предпосылки создания многопользовательской системы регистрации для разрабатываемых приложений 4

1.2Используемые технологии 4

1.3Создание системы аунтефикации 5

1.4 Создание системы сбора статистики и ее 9

1.5 Система контроля версий community-app 11

1.6 Структура проекта community-app 12

1.7 Описание технического оснащения команды разработчиков, и способов технического обслуживания используемого оборудования 14

1.6 Изучение вопросов охраны труда 18

Заключение 22

Список использованных источников 23

Изм.

Лист

№ докум.

Подпись

Дата

Лист

2

Разраб.

В.А Пантюхов

Пров.

И.В. Акиншева

Н. Контр.

Утв.

Отчет по преддипломной практике

Лит.

Листов

23

**МГУП, ИСиТ-151**

**Введение**

EPAM Systems — американская ИТ-компания, основанная в 1993 году. Крупнейший мировой производитель заказного программного обеспечения, специалист по консалтингу, резидент Белорусского парка высоких технологий. Штаб-квартира компании расположена в Ньютауне, штат Пенсильвания, а её отделения представлены в 25 странах мира.

Компания EPAM была основана в 1993 году двумя одноклассниками Аркадием Добкиным и Леонидом Лознером. Название компании происходило от «Effective Programming for America». Первые офисы были открыты в США и Белоруссии. Позже были открыты центральный североамериканский офис в Лоренсвилле, США, штат Нью-Джерси и центральный европейский офис в Будапеште, Венгрия, а также офисы по обслуживанию клиентов в Австрии, Австралии, Армении, Болгарии, Белоруссии, Великобритании, Германии, Индии, Ирландии, Казахстане, Канаде, Китае, Мексике, Нидерландах, ОАЭ, Польше, России, Сингапуре, Украине, Чехии, Швеции, Швейцарии.

**Деятельность**

1. ИТ-консалтинг
2. Разработка программного обеспечения
3. Интеграция приложений
4. Портирование и миграция приложений
5. Тестирование программного обеспечения
6. Создание выделенных центров разработки на базе EPAM Systems
7. Разработка цифровых стратегий

**Показатели деятельности**

Выручка компании за 2014 год составила 730,0 млн долл. Выручка за 2015 год составила 914,1 млн долларов, прибыль 106,0 млн долларов. Выручка за 2016 год составила 1,16 млрд долларов. Выручка за 2017 год составила 1,45 млрд долларов, чистая прибыль — 72,76 млн долларов при операционной прибыли в 172,9 млн долларов Выручка за 2018 год составила 1,84 млрд долларов, чистая прибыль — 240,3 млн долларов.

По итогам 2016 года EPAM занял 3-е место в рейтинге CNews «Крупнейшие ИТ-компании России 2016», а также 1-ое место среди разработчиков ПО в рейтинге РБК+ российских ИТ-компаний (3-е место в общем списке)

Предприятие занимается выпуском программного обеспечения и технологических решений для сторонних компаний (заказчиков).

**1 Индивидуальное задание**

**1.1 Предпосылки создания многопользовательской системы регистрации для разрабатываемых приложений.**

Приложение – Community-app было созданно в качестве платформы для регистрации новых приложений и предаствления им всем единого API для двухстороннего взаимодествия. Суть проекта заключается в том, пользователи могут запросить админа Community-app добавить свое приложение в платформу, которая предоставляет общую статистику и базу данных юзеров для всех приложений, зарегестрированных в нем.

Целью создания данного приложения заключалось в проведении образовательных соревнований(олимпиад) в реальном времени, с возможностью гибкой настройки времени, количество участников, а также построения архитектуры, в котором приложения, которое регистрирует в себя остальные приложения, является полностью независимой от них системой. Таким образом, приложения могут расширяться и модифицироваться независимо друг от друга, пока между ними существует единый интерфейс и унифицированная API – структура.

Основный задачи которые были решены: система аунтефикации пользователей; унификации общего интерфейса и API между приложениями, посредством создания и публикации собственной библиотеки; создания графической панели, для рисования графиков; Адаптация дизайна под мобильные телефоны и планшеты; сбор унифицированной статистики между приложениями, ее сохранение и обновление “на лету” для всех пользователей; создание системы разделения максимального количество участников для каждого приложения на отдельные комнаты со своим количеством участников и таймером; создание системы регистрации самих приложений через admin-панель.

**1.2 Используемые технологии**

При разработки данного приложения использовались следующие технологии. Общие: “concurrency” – консольное расширение для единовременного, последовательного исполнения нескольких команд из командной строки. Для сервера: @sendgrid/mail – пакет для автоматической отправки писем на почтовый ящик пользователя; axios – пакет для работы с стандартными API-запросами (GET, POST, UPDATE, DELETE); bcryptjs – пакет для шифровки данных; cors – пакет, предоставляющий middleware функциональность для express.js, которая позволяет предоставить доступ к ресурсам другого домена; express – фреймворк для создания web-приложений в среде исполнения node.js; inversify – пакет, предоставляющий возможность внедрения DI (Dependency Injection) для более логического разделения логики приложения. inversify-express-utils – расширение стандартного inversify для express.js фреймворка; inversify-inject-decorators – расширение стандартного inversify, предоставляющее декораторы для написания кода на typescript.js; inversify-logger-middleware – расширение стандартного inversify, предоставляющее middleware логику, работающую через express.js, реализующую функцию логирования API запросов и сохранения логов в отдельном файле; sequelize – ORM (объектно-реляционное отображение, или преобразование) для связи с базой данных Mysql; passport – пакет предоставляющий middleware для создания защищенных API в среде фреймворка express; validator – библиотека для валидации API запросов клиента и другие. Для клиента: React – библиотека, для построения SPA приложений; @material-ui/core, @material-ui/icons – пакеты для корректной работы библиотеки компонентов MeterialUiReact; d3 – Библиотека для визуального отражения статистической информации (например, построение графиков) работающая с svg; echarts - библиотека для работы с svg, предоставляющая готовые решения для визуального отображения данных; i18next, i18next-browser-languagedetector, react-i18next– пакеты для внедрения локализации на веб страницу в среде React. redux – библиотека, реализующая управление состоянием приложения в среде React; rxjs – библиотека, реализующая концепцию реактивного (посредством потоков) программирования, socket.io-client – библиотека, реализующая двухстороннее соединение с сервером и другие.

**1.3 Создание системы аунтефикации**

Для создания системы аунтификации пользователя использовалась библиотека passport, позволяющая легко добавить middleware для запросов. Система довольно простая: пользователь регистрируется, его данные добавляются в базу данных, при этому его пароль кодируется, после этого его перебрасывает на страницу логирования, где он вводит свои данные. Затем сервер возвращает его JWT-token, в котором зафишрованы данные, нужные клиенту (его имя, id) и тд. Клиент обнавляет состояние текущего пользователя, а сам токен помещает в cookie, для защиты от XSS атак. Так же у пользователя есть возмжность регистрации через Google.

Алгоритм регистрации можно описать следующей последовательностью:

1. Пользователь выбирает метод регистрации – через социальную сеть Google или Facebook, либо через email.
2. Пользователь заполняет все поля необходимые для регистрации (Рисунок 1)



Рисунок 1 – Форма регистрации

1. После нажатия кнопки регистрация, все данные будут проанализированы валидатором на клиенте. Если какие-то поля введены некорректно система оповестит об этом пользователя (рисунок 2)

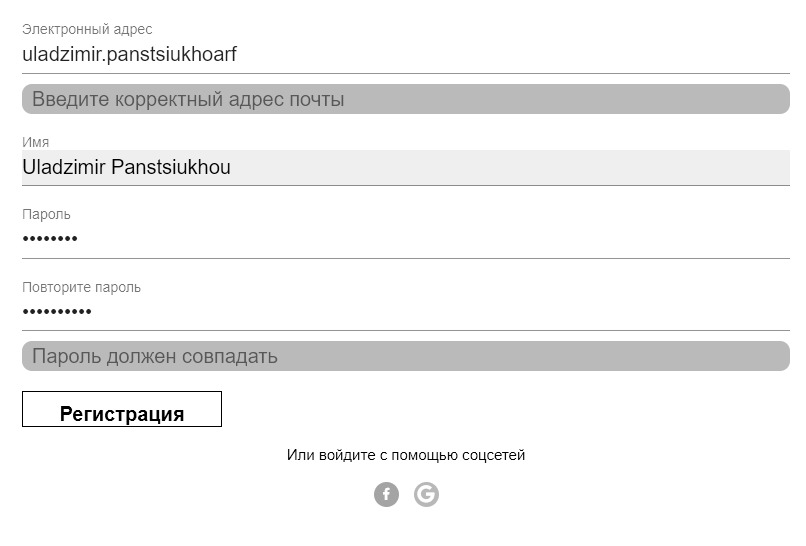


Рисунок 2 – Оповещения валидатора о неверности введенных данных

1. После исправления введенных данных, при нажатии кнопки регистрации будет отправлен запрос на сервер по адресу <http://localhost:3030/api/users/register> (Рисунок 3), где серверная программа провалидирует запрос еще раз. Зачем нужна двухстороння валидация? Дело в том, что сервер является независимо развернутым приложением. Другими словами, опытному пользователю совсем не обязательно отправлять запрос на сервер через интерфейс данной клиенской программы, он может сделать это в обход любых правил валидации, например через командную строку windows – смотрите листинг ниже.



Рисунок 3 – результат запроса на сервер

C:\WINDOWS\system32>curl -X POST http://localhost:3030/api/users/register

[{"code":1024,"msg":"Name must be between 2 and 30 characters"},

{"code":1001,"msg":"Name is required"},{"code":1010,"msg":"Email is Invalid"},

{"code":1000,"msg":"Email is required"},{"code":1003,"msg":"Password is required"},

{"code":1002,"msg":"Confirmed password is required"},

{"code":1025,"msg":"Password must be at least 6 characters and not more that 30"}]

Так же опытный пользователь может спровоцировать клик кнопки через консоль браузера, и запрос может быть отправлен с невалидными данными, если валидация обрабатывается не кликом, фрагмент данного кода приведен в листинге

temp1 = document.querySeelctorAll('.button-login')[0];

temp1.disabled = false;

temp1.click();

Существует еще много способов обойти валидацию клиента, и, кроме всего вышеперечисленного, некоторые данные возможно провалидировать только на сервере. Такими данными может быть, например, информация о том, зарегистрирован ли пользователь н момент оправления запроса или нет. В данном случаи юзер уже был зарегистрирован, о чем клиент успешно уведомил пользователя (Рисунок 4)

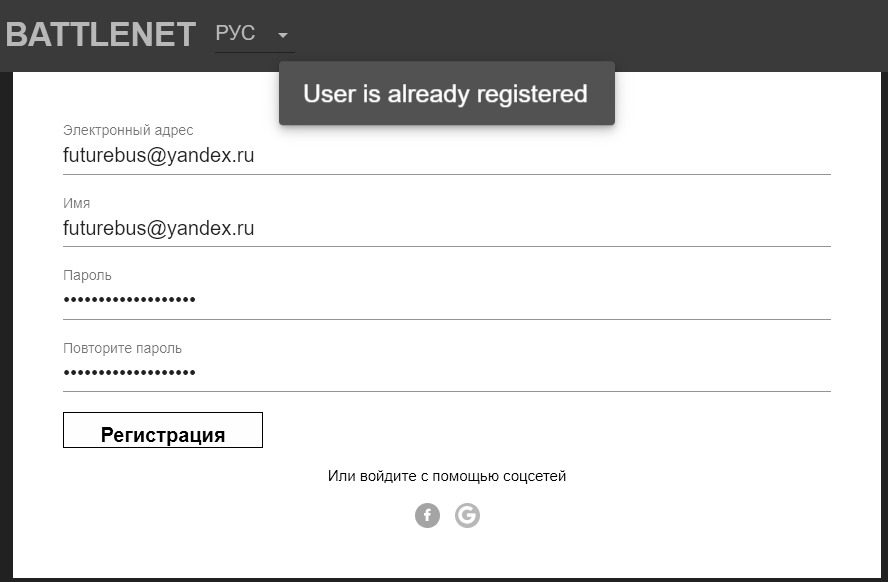


Рисунок 4 –Пользователь уже был зарегистрирован.

1. Повторная отправка формы с валидными данными, приведет к редиректу пользователя на защищенную ссылку <http://localhost:3000/#/battles>, где отображаются комнаты с текущими зарегистрированными приложениями (Рисунок 5)

Каждое приложение имеет команты, которые ожидают прихода достаточного числа пользователей, затем комнаты перенаправляют пользователей на другое приложение с помощью общего API (для этого была написана собственная библиотека – “community-layer”)

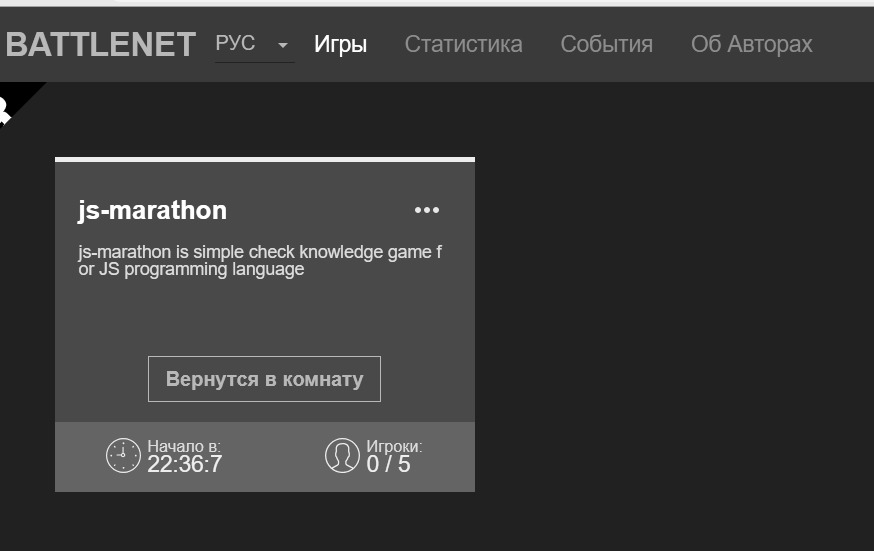


Рисунок 5 – Главная страница с созданными комнатами для зарегистрированных приложений.

**1.4 Создание системы сбора статистики и ее отображения**

Система собирает статистику с пользователей в том случаи если зарегистрированное приложение сделало соотвествующий API запрос с данными статистики и перенаправила пользователя обратно в платформу. Сама статистика представляет собой следующую композицию интерфейсов:

export interface GameData {

userToken: string;

playedTime: number;

scores: number;

resultStatus: ResultStatus;

participationStatus: ParticipationStatus;

}

export enum ResultStatus {

Init,

Win,

Lose,

Draw,

}

export enum ParticipationStatus {

Init,

Leave,

play,

}

где userToken – уникальный токен пользователя, который он имеет на время игры; playedTime – время игровой сессии; scores – количество набранных очков за время игровой сеcсии; resultStatus – статус пользователя на момент окончания игровой сесии (инициализирован, победил, проиграл, ничья); статус участия на момент окончания игровой сессии (инициализирован, вышел до окончания игровой сессии, был в игре на момент окончания игровой сесии). Данный интерфейс является общий для всех зарегистрированных приложений, так как он был унифицирован в общей библиотеке – community-layer.

После завершения игры зарегестрированное приложение делает API-request, который обрабатывает следующий код:

public async setGameResult(data: GameData[], appToken: string): Promise<boolean | ErrorBlock> {

const statistic = data;

try {

const tokenRow = await GamesModel.findOne({

where: { appToken }

});

const token = tokenRow && tokenRow.appToken;

if (token) {

let promises: Array<Promise<boolean>> = [];

// statistic = JSON.parse(statistic); // Uncomment to test with PostMan

promises = statistic.map((stat: Statistic) => this.saveStatistic(token, stat));

try {

await Promise.all(promises);

this.socketService.notifyAllClients('updateLeaders', tokenRow.appName);

return true;

} catch (error) {

throw error;

}

} else {

throw logicErr.notFoundAppToken;

}

} catch (error) {

if (error.code) {

throw error;

} else {

this.loggerService.errorLog(error);

throw technicalErr.databaseCrash;

}

}

}

В своей сигнатуре метод имеет ключевое слово async, которое помечает метод, как асинхронный. Выполнение метода становится микрозадачей (задачей, для решения которой, нужно дождаться ответа от базы данных или другого сервера). Микрозадача будет добавлена в основной стэк выполнения только после получения вышеупомянутого ответа (т.е. будут получены так называемые “отложенные ресурсы”). То есть основной поток выполнения не будет загружен до получения ответа от стороннего сервиса, и несколько пользователей могут продолжать работу с приложением параллельно. Такое поведение возможно благодаря встроенному в среду node.js движку V8. Синхронизацией очереди задач, микрозадач и стэка занимается Event Loop.

Метод принимает данные о результатах игре и также токен этой игры. Токен приложение выдается, при его регистрации через admin-панель, которая является независимо развернутым приложением.

Далее метод saveStatistic сохраняет статистику для каждого пользователя, а затем все пользователи получают обновленные данные через метод notifyAllClients в реальном времени обновляя состояние клиента.

Если какой-то из асинхронных методов, например, обращение к базе данных, возвращает ошибку, то выполнение кода переходит в метод catch, который генерирует соответствующую ошибку и записывает ее в лог ошибок.

Сама обработка ошибок в данном приложением имеет свою архитектуру и нуждается в отдельном описании.

Контроллер так же имеет функциональность по сортировки лучших пользователей, последних сыгран игр по каждому определенному пользователю, наиболее популярных игр и лидеров текущий игровой сессии.

Для отображения статистики на клиенте используется графическая панель для рисования графиков, основная логика которой инкапсулирована в стороннем решении.

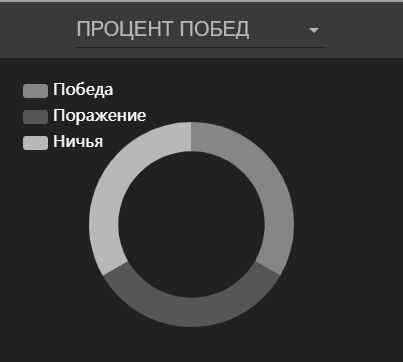


Рисунок 6 – Графическая панель

**1.5 Система контроля версий community-app**

Работа над данным приложением в настоящее время все еще ведется со стороны “ЭПАМ Системз”. В качестве системы для контроля версий используется Github.

GitРub (рисунок 3) – крупнейший веб-сервис для хостинга IT-проектов и их совместной разработки.

Веб-сервис основан на системе контроля версий Git и разработан на Ruby on Rails и Erlang компанией GitHub, Inc. Сервис бесплатен для проектов с открытым исходным кодом и (с 2019 года) небольших частных проектов, предоставляя им все возможности (включая SSL), а для крупных корпоративных проектов предлагаются различные платные тарифные планы.

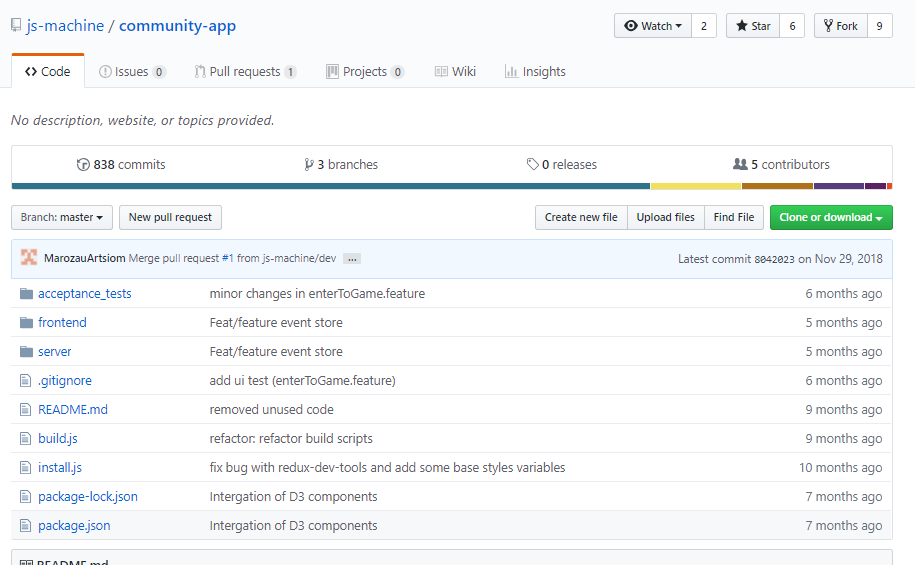


Рисунок 7 ­– Интерфейс GitHub

**1.6 Структура проекта community-app**

Структура проекта представлена на рисунке 8

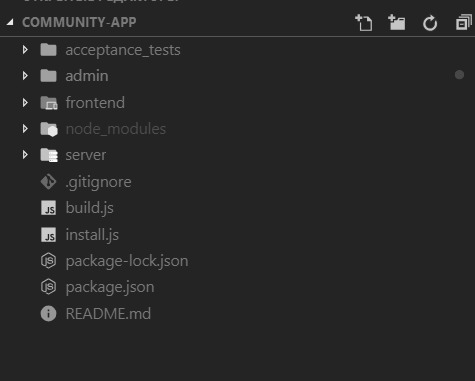


Рисунок 8 – Структура проекта

В папке admin размещаются файлы независимого приложения, которое имеет свой клиент и сервер и разворачивается на другом хосте.

acceptance\_tests – в этой папке размещены тесты для проверки работоспособности приложения. Frontent/server – папки содержащие клиент и сервер самого community-app. node\_modules – стандартная папка сгенерированная npm при установке зависимостей и подзависимостей, деревовидная структура которых описана в файлах package.json и package.lock.json. Файлы install.js и build.js – скрипты, позволяющие собрать и развернуть проект.

При первом запуске серверного приложения seq uilize “на лету” сгенерирует необходимые таблицы в базу данных. Описание таблиц (их поля и взаимосвязи) описываются специальными моделями. Например следующий код, при создании таблиц user, roles установит между ними связь “многие ко многим” Рисунок. Таким образом, можно “на лету” создавать и связывать таблицы, данный код приведен в листинге ниже.

export const makeAssociations = () => {

UserRoles.hasMany(UserModel, {

as: 'users',

foreignKey: 'id',

foreignKeyConstraint: true

});

UserRoles.hasMany(RoleModel, {

as: 'roles',

foreignKey: 'id',

foreignKeyConstraint: true

});

};

После добавления статистики, клиент автоматически, без перезагрузки страницы, уведомит всех пользователей. Это возможно благодаря технологии Socket, которая позволяет автоматически создать непрерывную связь между клиентом и сервером для каждого отдельного пользователя. С программной точки зрения эта логика была инкапуслирована в отдельный сервис, позволяющий генерировать события на сервере и уведомлять об их наступлении всех пользователей, которые в данный момент находятся на сайте, без перезагрузки страницы, код данного метода приведен в листинге ниже.

public notifyAllClients(eventName: string, payload: RoomInfo[] | string): void {

this.clients.forEach((client) => {

client.emit(eventName, payload);

});

}

Благодаря этому сервису, данные всех пользователей, представление этих данных на клиенте, и данные на сервере полностью синхронизированы и обновляются в реальном времени. Таким образом, статистика зарегистрированных приложений обновляется в реальном времени, что создает конкурентную среду для каждого отдельного пользователя. Этот механизм позволяет проводить соревнования на зарегистрированных приложениях. Такие тестовые соревнования уже были проведены на JS Meetup в 2018 году в Могилеве, в тестировании принимали участие более 100 человек одновременно. Максимальное количество участников зависит от настройки, характеристик и количество серверов, на которых развернуто приложение.

**1.7 Описание технического оснащения команды разработчиков, и способов технического обслуживания используемого оборудования**

Работа программиста предполагает работу за персональным компьютером. Техническое оснащение отдела программирования можно описать, описав рабочее место программиста. Оно в основном состоит из:

1 Источник бесперебойного питания;

2 Компьютерная мышь;

3 Клавиатура;

4 Монитор;

5 Системный блок;

5.1 Жесткий диск, твердотельный накопитель;

5.2 Материнская плата;

5.3 Процессор;

5.4 Блок питания;

5.5 Оперативная память;

5.6 Видеокарта;

6 Принтер.

Рабочие места не являются шаблонными. Оборудование по мере выхода из строя заменяется новым, если его невозможно отремонтировать, а также при моральном устаревании так же заменяется новым, по возможности. В связи с этим техническое оснащение отдела весьма различно и меняется от места к месту.

Источники бесперебойного питания APC. Используются различные ИБП, основная функция которых – это предотвратить отключение компьютера при потере электричества. Не все рабочие места оборудованы ИБП. Используется серия Back-UPS (рисунок 5). Цены от 216 белорусских рублей. Четыре выходных розетки, три розетки с резервным питанием. Время автономного питания от 3,8 минут до 7 минут. Мощность 210 – 300 Вт. Полная мощность 350 – 500 VA. Выходное напряжение – 230 В. Максимальный входной ток – 7 А. С USB интерфейсом, батарея свинцово-кислотная.



Рисунок 9 – Источник бесперебойного питания

Компьютерные мыши. Нельзя выделить какую-то преобладающую группу одного производителя. Некоторые пользователи используют достаточно специфические мыши с различными форм-факторами. Поэтому за основу я возьму тот факт, что используются обычные мыши с тремя кнопками.

Исходя из этого, примем тот факт, что используются оптические мыши с тремя кнопками и возможностью нажатия на среднюю кнопку (колесико), проводные. Самым распространенным примером является мышь Logitech B100 ценой от 10 рублей с оптическим сенсором 800dpi (рисунок 6). Данный вид оборудования является расходным материалом, поэтому нельзя четко выделить какой-то один вид и тип мышей.



Рисунок 10 – Мышь

Клавиатуры. С клавиатурами дела обстоят примерно так же, как и с мышками. Каждый использует ту клавиатуру, которая ему нравится и подходит для работы. В основном используются стандартные мембранные клавиатуры.

Примем тот факт, что используются офисные мембранные клавиатуры, проводные с USB интерфейсом подключения (например, Logitech classis keyboard K100 ценой от 20р, находящаяся на рисунке 7). Данный вид оборудования является расходным материалом, поэтому нельзя четко выделить какой-то один вид и тип клавиатур.



Рисунок 11 – Клавиатура

Мониторы. В отделе используются мониторы фирмы DELL. На некоторых рабочих местах мониторов два, и закреплены они фирменным кронштейном, образуя тем самым единое рабочее пространство из двух мониторов.

Монитор DELL серии Ultrasharp (U2515H). Цена от 900 белорусских рублей. Диагональ ~ 25 дюйма. ЖК матрица. Соотношение сторон – 16:9. IPS матрица, разрешение экрана 2560х1440. Яркость 350 кд/м2ю, контрастность 1000:1, время отклика 6 мс, usb 3.0, hdmi, displayport. Пример представлен на рисунке 8.



Рисунок 12 – Монитор

Кронштейн Dell Dual Monitor Arm (рисунок 9). Настольный кронштейн для мониторов черного цвета, с углом наклона 60-40 градусов и углом поворота 180 градусов. Цена от 730 белорусских рублей.

Рисунок 13 – Кронштейн Dell Dual Monitor Arm

Так же, рассматривая мониторы можно отметить, что некоторое количество моноблоков, которые используются взамен устаревших и сломавшихся системных блоков, и мониторов (моноблок - это системный блок и монитор в едином корпусе).

Принтеры и МФУ. Используются принтер фирмы Ricoh (SP 220Nw). Лазерные, черно-белая печать форматов А4. Цены начинаются от 215 белорусских рублей. Пример расположен на рисунке 10). Скорость печати 23 страниц в минуту, поддерживается операционными системами семейства Windows. Вместимость входных лотков 150 листов, ресурс картриджа 700 листов. Присутствуют беспроводные интерфейсы (Wi-Fi).

HP LaserJet Pro M1217nfw (рисунок 99). Цена неизвестна, снят с производства. Печать черно-белая, лазерная. Скорость – 18 листов в минуту. Копирование, сканер, факс. Поддерживается ОС семейства Windows. Возможна печать с мобильных устройств. Максимальная месячная нагрузка – 8000 страниц в месяц. Присутствуют беспроводные интерфейсы связи (Wi-Fi).

Рисунок 14 – Принтер и МФУ

Операционные системы и прочий софт. На рабочих станциях установлены лицензионные версии операционной системы Windows 10 x64. Так же установлены лицензионные версии Microsoft Office. Всё программное обеспечение, которое используется в работе отдела так или иначе лицензировано.

Системный блок. Состоит из комплектующих, такие как: жесткий диск, твердотельный накопитель (HDD и SSD); материнская плата; процессор; блок питания; оперативная память и видеокарта.

Так как оборудование устаревает, выходит из строя или происходит замена старого оборудования на новое, то состав системных блоков от рабочего места к рабочему месту может не совпадать, поэтому, руководствуясь системными требованиями программного обеспечения, которое использует в своей работе отдел программирования, можно сделать вывод, что:

– Жесткий диск не менее 500 Гбайт;

– Твердотельный накопитель не менее 120 Гбайт;

– Процессор с тактовой частотой не ниже 1,8 ГГц, не менее двух ядер;

– Не менее 4 Гбайт оперативной памяти;

– Видеокарта с разрешением, которое может поддерживать монитор;

– Материнская плата, поддерживающая интерфейсы HDD и SDD, процессора, ОЗУ, видеокарты.

Известно, что используются готовые компьютеры DELL на плате Precission 3630 с характеристиками (рисунок 11). Цена от 2800 белорусских рублей:

– Процессор Intel Core I7 8700, 6 ядер, тактовая частота 3,2 ГГц;

– SDD – 512 Гбайт;

– HDD – 1 Тбайт;

– Оперативная память 16 Гбайт;

– Intel UHD Graphics 630 и AMD Radeon WX4100.

Условия охраны труда и техники безопасности полностью соответствуют СанПиН «Требования при работе с видеодисплейными терминалами и электронно-вычислительными машинами», так как основной вред идет именно от них.



Рисунок 15 – Компьютер DELL

**1.8 Изучение вопросов охраны труда**

Современный офис тяжело представить без компьютера. Тем более, если речь идет об IT-компании, осуществляющей деятельность в области программного обеспечения.

Основными нормативными правовыми актами по охране труда при работе с компьютерами являются:

– Типовая инструкция по охране труда при работе с персональными электронными вычислительными машинами, утвержденная постановлением Минтруда и соцзащиты Республики Беларусь от 24.12.2013 № 130;

– Межотраслевые общие правила по охране труда, утвержденные постановлением Минтруда и соцзащиты Республики Беларусь от 03.06.2003 № 70 (гл. 17 «Электробезопасность»);

– Санитарные нормы и правила «Требования при работе с видеодисплейными терминалами и электронно-вычислительными машинами», Гигиенический норматив «Предельно допустимые уровни нормируемых параметров при работе с видеодисплейными терминалами и электронно-вычислительными машинами», утвержденные постановлением Минздрава Республики Беларусь от 28.06.2013 № 59 (далее – СанПиНиГН № 59);

– ТКП 181-2009;

– ТКП 427-2012;

– ГОСТ 12.1.030-81 «Система стандартов безопасности труда. Электробезопасность. Защитное заземление, зануление», введенный в действие на территории Республики Беларусь постановлением Госстандарта Республики Беларусь от 17.12.1992 № 3 (далее – ГОСТ 12.1.030-81).Основная цель организации в области охраны здоровья и безопасности   
труда – это обеспечение безопасности и сохранение здоровья персонала.

Основные направления политики в области охраны здоровья и безопасности труда представлены ниже:

1. Соблюдать законодательные и другие требования в области охраны здоровья и безопасности труда, применимые к деятельности организации;
2. Внедрять мероприятия, направленные на улучшения условий труда и повышение уровня безопасности труда, снижение риска производственных травм и профессиональных заболеваний;
3. Постоянно проводить анализ, совершенствовать результативность системы менеджмента охраны здоровья и безопасности труда, представлять необходимые средства и ресурсы для обеспечения функционирования системы;
4. Проводить работу по определению, оценке, управлению и снижению рисков в области охраны здоровья и безопасности труда;
5. Привлекать персонал организации к активному участию по реализации целей в области охраны здоровья и безопасности труда;
6. Совершенствовать технологические процессы, внедрять ресурсосберегающие и энергосберегающие технологии, стремиться к рациональному использованию материальных и энергетических ресурсов;
7. Постоянно повышать квалификацию персонала организации в области охраны здоровья и безопасности труда;
8. Вести открытый диалог с персоналом организации и общественностью о своей деятельности в области охраны здоровья и безопасности труда;
9. Доводить требования политики в области охраны здоровья и безопасности труда до сведения персонала организации, подрядчиков, всех заинтересованных сторон.

Так же, учитывая отдел программирования, в котором находится большое количество компьютеров, то он так же представляет опасность для работника при несоблюдении правил охраны труда.

В частности, имеются опасности и вредности при работе с ЭВМ, в том числе и с портативными (ноутбуки и др.), а также с периферийными устройствами (принтеры, сканеры и др.).

К помещениям такого рода применяются различные требования, в которых должны соблюдаться требования по микроклимату в рабочей зоне, освещению, организации и оборудованию рабочего места, организацию медицинского обслуживания пользователей ЭВМ, а также режимов труда и отдыха.

Поэтому, необходимо руководствоваться СанПиН «Требования при работе с видеодисплейными терминалами и электронно-вычислительными машинами» при устройстве рабочего места, работе за персональным компьютером и следить за режимом труда и отдыха, посадкой и другими важными параметрами.

Конструкция компьютера, дизайн и совокупность эргономических параметров должны обеспечивать надежное и комфортное считывание отображаемой информации в условиях эксплуатации:

– конструкция компьютера должна обеспечивать возможность поворота корпуса в горизонтальной и вертикальной плоскостях с фиксацией в заданном положении для обеспечения фронтального наблюдения экрана;

– дизайн компьютера должен предусматривать окраску корпуса в спокойные мягкие тона с диффузным рассеиванием света, т.е. рассеиванием света во всевозможных направлениях;

– корпус компьютера должен иметь матовую поверхность с коэффициентом отражения (0,4–0,6) и не иметь блестящих деталей, способных создавать блики;

– конструкция компьютера должна предусматривать регулирование яркости и контрастности.

В организации должна иметься документация по эксплуатации новых (модернизированных) компьютеров.

Компьютеры подлежат государственной санитарно-гигиенической экспертизе. Реализация и эксплуатация новых (модернизированных) компьютеров на территории Республики Беларусь без санитарно-гигиенического заключения не допускаются.

Помещения для эксплуатации компьютеров должны иметь естественное и искусственное освещение.

Запрещается выполнение основной работы с использованием компьютеров на постоянных рабочих местах без естественного освещения, если это не обусловлено технологическим процессом.

Основная работа – работа, которая занимает не менее 50 % времени в течение рабочего дня (смены).

Постоянное рабочее место – место, на котором работающий находится большую часть своего рабочего времени (более 50 % или более 2 ч непрерывно).

Естественное освещение на рабочих местах с компьютерами должно осуществляться через световые проемы, ориентированные преимущественно на север, северо-восток, восток, запад или северо-запад.

Оконные проемы должны быть оборудованы регулируемыми устройствами типа жалюзи, занавесей, внешних козырьков и др.

Площадь одного рабочего места для пользователей компьютеров на базе ЭЛТ должна составлять не менее 6 м2.

Минимальная площадь одного рабочего места для взрослых пользователей и обучающихся учреждений профессионально-технического, среднего специального и высшего образования с использованием компьютеров на базе ЭЛТ может составлять не менее 4,5 м2 при следующих условиях:

– отсутствие на рабочем месте периферийных устройств (принтер, сканер и др.);

– продолжительность работы должна составлять не более 4 ч в день.

Площадь одного рабочего места для пользователей компьютеров на базе плоских дискретных экранов (жидкокристаллические, плазменные и др.) должна составлять не менее 4,5 м2.

При возведении и реконструкции зданий с помещениями для компьютеров эти помещения следует проектировать высотой от пола до потолка не менее 3,0 м.

Помещения, где размещаются рабочие места с компьютерами, должны быть оборудованы защитным заземлением (занулением) в соответствии с техническими требованиями по эксплуатации и ГОСТ 12.1.030-81.

Запрещается размещать рабочие места с компьютерами на расстоянии менее 10 м от силовых кабелей, вводов и высоковольтных трансформаторов.

Помещения, в которых для работы используются преимущественно компьютеры (диспетчерские, операторские, расчетные, классы и др.), не должны граничить с помещениями, в которых уровни шума и вибрации превышают нормируемые значения для данной категории проводимых в них работ и их типа (механические цеха, мастерские, гимнастические залы и др.) согласно Санитарным нормам № 115.

**Заключение**

В ходе технологической практики я: изучил профессиональные системы, применяемые на в организации, платформы и технологии; изучил техническое оснащение отдела программирования, а так же способы технического обслуживания используемого оборудования; разработал круг задач, решение которых предполагается реализовать в процессе работы над дипломным проектом; изучил вопросы охраны труда и экологии; изучил технико-экономические показатели.

Изучил в практических условиях приемы и принципы разработки программного обеспечения в отделе программирования. Так же провел технико-экономическое обоснование необходимости внедрения разрабатываемого мной программного продукта. Путем изучения характеристик процесса разработки отдельных программных продуктов, для обоснования его дальнейшего совершенствования по ряду показателей, а также ознакомился с ведением организационно-распорядительной документацией и нормативными документами по эксплуатации и ремонту оборудования, содержанию и периодичности обслуживания и ремонта. Уделил внимание вопросам взаимодействия с другими службами предприятия при эксплуатации сетевых ресурсов и изучил организационную структуру предприятия, отдела программирования.

В ходе практики мной были изучены аспекты разработки программных модулей для дипломного проектирования.

В ходе преддипломной практики индивидуальное задание было выполнено в полном объеме. Разработанное приложение отвечает всем современным требованиям безопасности и скорости работы. Данная платформа может быть использовована для передачи данных о зарегестрированных приложениях в общую систему, с возможностей добавления, удаления и изменения новых приложений. Любое изменение публичных данные на сервере моментально синхранизируется со всеми клиентами. Приложение было протестировано в реальных условиях, при большом количестве клиентов.

**Список использованных источников**

1 СТП СМК 4.2.3-01-2011 Общие требования и правила оформления текстовых документов.− Могилев: Ризограф УО МГУП, 2011. ‑ 47 с.

2 ГОСТ 19.701-90. ЕСПД. Схемы алгоритмов, программ, данных и систем. Обозначения условные и правила выполнения.

3. Климов, Александр JavaScript на примерах / Александр Климов. - М.: БХВ-Петербург, 2009. - 379 c.

4. Пауэрс, Ш. Изучаем Node. Переходим на сторону сервера /Ш. Пауэрс. - М.: Питер, 2017. - 304 c.

5. Чекко, Рафаэлло Графика на JavaScript / Рафаэлло Чекко. - М.: Питер, 2013. - 353 c.