Рисунок 1 – Главная страница сайта

Сегодня для любого системного администратора актуальны вопросы касательно техники в их организации, в связи с чем, можно уверенно сказать что данная разработка будет полезным источником информации о компьютерной техники в организации.

1. Алексеев А.. Введение в Web-дизайн. Учебное пособие. — М.: ДМК Пресс, 2019. — 184 с.
2. Гарретт Д. Веб-дизайн. Элементы опыта взаимодействия / Д. Гарретт. — СПб.: Символ-плюс, 2016. — 192 с.
3. Дакетт Д. HTML и CSS. Разработка и дизайн веб-сайтов. — М.: Эксмо, 2019. — 480 с.
4. Диков А. В. Клиентские технологии веб-дизайна. HTML5 и CSS3. Учебное пособие. — М.: Лань, 2019. — 188 с.
5. Дэвид Макфарланд. Новая большая книга CSS. — М.: Питер, 2018. — 720 с.
6. Кирсанов Д. Веб-дизайн: книга Дмитрия Кирсанова / Д. Кирсанов. — М.: Символ, 2017. — 368 с.
7. Киселев С.В. Веб-дизайн / С.В. Киселев. — М.: Academia, 2019. — 285 с.

Шевелев А.М., Рысин М.Л.

Практические приёмы создания виртуального коворкинга для разработчиков программного обеспечения

*РТУ МИРЭА
(Россия, Москва)*

doi: 10.18411/lj-07-2021-15

Аннотация

В статье рассматриваются этапы создания элементов виртуального коворкинга для разработчиков программного обеспечения. Авторами была показана структура типовой организации, разрабатывающей программное обеспечение, проанализированы требования к системе виртуального коворкинга, спроектирована архитектура программной системы, реализованы её компоненты.

Ключевые слова: виртуальный коворкинг, реактивное программирование, клиент-серверная архитектура, реляционные базы данных.

Abstract

The article discusses the stages of creating the virtual coworking elements for software developers. The authors showed a structure of a typical software developing company, analyzed the requirements for the virtual coworking system, designed the program system architecture, created its components.

Keywords: virtual coworking, reactive programming, client-server architecture, relational databases.

На сегодняшний день невозможно представить жизнь современного человека и в России и за её пределами без доступа к сети Интернет [1]. Оценить огромную роль Интернета смог каждый житель Земли в связи с наступлением пандемии COVID-19. Большой процент работающего населения был переведен в удаленный режим работы, студенты и школьники – на дистанционное обучение с использованием средств аудио- и видео- общения (Zoom, Skype, Google Meet).

Свои новшества и в современной организации бизнеса. Многие компании переоборудуют свои офисы в openspace-пространства, возрос процент коворкингов – оборудованных помещений с необходимой инфраструктурой, где можно арендовать рабочее место для учебы или работы на необходимое время.

К сожалению, пандемия затянулась, и ограничения, связанные с эпидемиологической обстановкой, в той или иной мере, остаются с человечеством до сих пор. В среде IT-разработки это может привести к увеличению сроков, и как следствие, увеличению стоимости продукта.

Объективно обусловлена потребность в виртуальной среде, которая будет считаться некоторой социальной сетью для программистов, и будет содержать необходимую функциональность для совместной комфортной разработки ПО.

В состав организации, профессионально занимающейся разработкой сложного ПО, входит множество людей.

Часто структура типовой команды разработчиков ПО состоит из менеджера проекта, команды дизайнеров, команды frontend и backend разработчиков, команды инженеров по обеспечению качества.

Модель бизнес-процесса «Создание программного обеспечения» в нотации IDEF0 представлена на рис. 1.

Рис. 1 – бизнес процесс «Создание ПО». Верхний уровень.

Декомпозиция её верхнего уровня состоит из следующих блоков, которые являются этапами жизненного цикла ПО (рис. 2): формирование требований к ПО, проектирование, реализация, тестирование, эксплуатация и сопровождение.

Рис. 2 – Декомпозиция бизнес-процесса «Создание ПО»

На основе полученных диаграмм можно заметить, что работники такой компании находятся постоянно в контакте с коллегами как из своего отдела, так и из соседнего.

Например, дизайнер может внести правки в макет приложения после проведения юзабилити-тестирования, и фронтенд-разработчик должен узнать об этом как можно скорее. Также менеджер проекта должен участвовать в управлении проектом на всех этапах жизненного цикла программы, начиная от формирования требований до внедрения в эксплуатацию и последующей поддержки приложения, чтобы команда успела сделать приложение в срок, оговоренный заказчиком и удостоверенным в техническом задании.

Внедрение разрабатываемого приложения позволит избежать временных простоев при коммуникации разработчиков.

В настоящий момент на рынке существует ограниченное количество продуктов, помогающих разработчикам взаимодействовать между собой – GitHub, JsFiddle, Яндекс.Переговорки, StackOverFlow. Каждое из таких решений обладает своими достоинствами и недостатками, анализ которых приводит к выводу, что не существует программной системы, полностью покрывающей требования предприятия, и соответственно возникает необходимость в новых программных разработках в этой сфере.

Согласно определению, данному Иеном Соммервилем и Питом Сойером, требование – это спецификация того, что должно быть реализовано в рамках программы. Требования являются ограничениями в процессе разработки системы, в них описаны её поведение, свойства и атрибуты [2].

В результате анализа требований получают спецификации разрабатываемого ПО, которые содержат постановку решаемых задач, уточняют их взаимодействие и являются основой для создания технического задания. Существует два подхода к построению моделей этапа анализа и определения спецификаций: структурный и объектно-ориентированный подход [3,4].

В рамках последнего представлены спецификации в форматах диаграммы прецедентов (рис. 3) со сценариями и диаграммы деятельности для варианта использования «Создание чата» (рис. 4).

Рис. 3 – Диаграмма прецедентов предметной области

Анализ требований, в свою очередь, позволил определить в техническом задании, что разрабатываемая система должна являться приложением с клиент-серверной архитектурой.

Для реализации клиентской части программной системы должны быть использованы средства, отвечающие современным стандартам W3C: язык гипертекстовой разметки HTML5, язык разметки каскадной таблицы стилей CSS3, библиотека реактивных компонентов React, фреймворк Bootstrap.

Рис. 4 – Диаграмма деятельности для варианта использования «Создание чата»

Вся нагрузка по обработке бизнес-логики, хранению пользовательских данных, обеспечении аутентификации и регистрации ложится на сервер, который представляет собой REST API, написанное на Node.js с помощью фреймворка Express.js.

База данных сайта – служит для хранения данных пользователей и представима набором SQL-скриптов [4].

В результате исследования предметной области базы данных были выделены следующие сущности: Пользователь; Участник; Устройство; Доступ; Список_блокировки; Диалог; Удаленный диалог; Удаленное сообщение; Сообщения; Контакты. На рис. 5 представлена соответствующая даталогическая модель этой базы данных.

В ходе выполнения работы мы будем использовать следующий инструментарий: Ramus для моделирования бизнес-процессов, Figma для проектирования макетов клиентской части и Microsoft Visual Studio Code для написания программного кода.

На основании даталогической модели были созданы скрипты создания таблиц, реализованные на PostgreSQL. В базе данных используются все три основных вида связи. Для примера рассмотрим процесс создания таблицы «Пользователь» (листинг 1), и её заполнение (листинг 2).

Рис. 5 – Даталогическая модель базы данных

Листинг 1

SQL-скрипт создания таблицы:

```
CREATE TABLE "Пользователь"(  
  ID INT,  
  phone VARCHAR(16),  
  email VARCHAR(255),  
  password VARCHAR(40),  
  first_name VARCHAR(20),  
  last_name VARCHAR(20),  
  middle_name VARCHAR(20),  
  verification_code VARCHAR(6),  
  is_active BOOL,  
  is_reported BOOL,  
  created_at TIMESTAMP,  
  updated_at TIMESTAMP,  
  PRIMARY KEY(ID)  
)
```

Листинг 2

SQL-скрипт вставки данных в таблицу:

```
INSERT INTO Пользователь VALUES(1, '79999999', 'sh@ya.ru', '12345', 'Alex',  
'Shevelev', 'Olegovich', '220907', true, false, now(), now())  
INSERT INTO Пользователь VALUES(3, '7777777', 'student@mirea.ru', '45qwwe',  
'Polina', 'Ivanova', 'Valerevna', '909087', true, false, now(), now())  
INSERT INTO Пользователь VALUES(3, '7777777', 'student@mirea.ru', '45qwwe',  
'Polina', 'Ivanova', 'Valerevna', '909087', true, false, now(), now())
```

Клиентская часть реализована с помощью UI-библиотеки компонентов React. React имеет обширную экосистему, которая состоит из самого React, React-DOM для управления виртуальным DOM'ом, React-Router, отвечающий за маршрутизацию.

Язык разметки JSX – это расширение языка гипертекстовой разметки HTML, который более похож на смесь HTML и XML.

Проект Single Page Application будет использовать библиотеку реактивных компонентов React и реализовывать Flux-архитектуру благодаря использованию библиотеки Redux [5].

Рассмотрим компоненты более подробно.

Так, например, компонента Dialogs состоит из компонент Dialog, представляющего список чатов с пользователями, и Message, представляющего конкретный диалог с конкретным программистом (рис. 6).

Рис. 6 – Компонента Dialogs

Обратим внимание на Листинг 3.

Листинг 3

Компонента «Dialogs»:

```
import React from 'react';
import Dialog from './Dialog/Dialog';
import s from './Dialogs.module.css';
import Message from './Message/Message';

const Dialogs = (props) => {
  let dialogsArray = props.state.dialogsPage.dialogs.map((d) => <Dialog id={d.id}
name={d.name} />);
  let messagesArray = props.state.dialogsPage.messages.map((m) => <Message
text={m.message} />);

  let messageRef = React.createRef();

  const addMessage = () => {
    props.addDialogMessage(messageRef.current.value);
    messageRef.current.value = "";
  }

  const changeMessageHandler = () => {
    props.changeDialogMessage(messageRef.current.value);
  }

  return <div className={s.dialogs_wrapper}>
    <div className={s.dialogs}>
      {
        dialogsArray
      }
    </div>
    <div className={s.messages}>
      {
        messagesArray
      }
    </div>
    <textarea ref={messageRef} value={props.state.dialogsPage.newMessage}
onChange={changeMessageHandler}></textarea>
    <button onClick={addMessage}>Отправить</button>
  </div>
}
export default Dialogs;
```

Используя Node.js и фреймворк Express.js, можно реализовать API в RESTful-подходе для взаимодействия с пользователем. REST-стиль особенно удобен при создании Single Page Application.

В качестве примера разберём функцию (Листинг 4), которая отправляет запрос к базе данных и получает все объекты.

Листинг 4

Вывод всех пользователей:

```
async function GetUsers() {
  const response = await fetch("/dialogs/users", {
```



```

method: "GET",
headers: { "Accept": "application/json" }
});
if (response.ok === true) {
  const users = await response.json();
  let rows = document.querySelector("tbody");
  users.forEach(user => {
    rows.append(row(user));
  });
}

```

Протестируем полученную программную систему на быстродействие, отклик системы и соответствие семантическим нормам. В этом нам помогут сервисы Google PageSpeed Insights и W3C Validator.

Проведенный анализ показал, что, несмотря на небольшие недочеты, наше веб-приложение загружается максимально быстро (рис. 7). Например, время, в течение которого страница становится полностью готова к взаимодействию с пользователем равняется меньше одной секунды, что является успешным показателем.

Рис. 7 – Тестирование в Google PageSpeed Insights

Проверка разметки – важный шаг к обеспечению технического качества веб-страниц. Консорциум World Wide Web разработал специальный валидатор (рис. 8), который позволяет проверять HTML разметку на предмет семантических норм [6].

Рис. 8 – Тестирование в W3C Validator

Как можно заметить, валидатор не заметил серьезных проблем в программной системе и советует обрамлять тег `<section>` одним из заголовков `<h2-h6>`.

С тестированием на предмет соответствия современной семантической верстке и проверкой быстродействия программная система справилась отлично.

Таким образом, получена клиент-серверное решение для виртуального коворкинга, предназначенное для повышения эффективности разработки ПО.

1. Г.И. Абдрахманова, О.Е. Баскакова, К.О. Вишневецкий, Л.М.Голхберг и др / Тенденции развития интернета в России и зарубежных странах: аналитический доклад [Электронный ресурс]. URL: <https://issek.hse.ru/mirror/pubs/share/345060549.pdf> (дата обращения 12.06.2021).
2. Вигерс К., Бигги Д. Разработка требований к программному обеспечению – СПб: «Русская редакция», 2019 – 736 с.
3. Забродин, А. В. Основы проектирования информационных систем с помощью языка UML: учебное пособие / А. В. Забродин, В. П. Бубнов. — СПб.: ПГУПС, 2018. — 46 с.
4. Рудаков А.В., Федорова Г.Н. Технология разработки программных продуктов. Практикум: учебное пособие для студентов. – Москва: «Академия», 2017 – 192 с.
5. Хортон, А. Разработка веб-приложений в ReactJS / Перевод с английского Р. Н. Рагимова. — Москва: ДМК Пресс, 2016. — 254 с.
6. Валидатор W3C [Электронный ресурс] URL: <https://developers.google.com/speed/docs/insights/v5/about?hl=ru> (дата обращения 12.06.2021).