WEB SAYTLARNI HIMOYALASHDA PAROLLARNI GENERATSIYA QILUVCHI TIZIM

S. N. Djumayev
Toshkent axborot texnologiyalari universiteti
Samarqand filiali o‘qituvchisi
sindordjumayev@gmail.com
D. G‘. Yusupov
Toshkent axborot texnologiyalari universiteti
Samarqand filiali talabasi
dilshodyusupov1998.03.14@gmail.com
Sh. S. Anorboyev
Toshkent axborot texnologiyalari universiteti
Samarqand filiali talabasi
shoxruhanorboyev0708@gmail.com

Annotatsiya: *Ushbu maqolada web saytlarni himoyalashda parollarni avtomatik generatsiya qilib beruvchi tizim yaratish masalasi keltirilgan. Yaratilgan tizim telegram bot shaklida bo‘lib, foydalanuvchiga ma’lum vaqt oralig‘ida parollarni generatsiya qilish orqali yangilab xabar junatib turadi.*

Kalit so‘zlar: *Sayt, FrontEnd, parol, generatsiya, React, Deklarativ, JavaScript, XML, class.*

Ushbu maqolamizda biz parollarni generatsiya qiluvchi tizim yaratish uchun biz telegram botdan foydalanamiz. Bu botdan keyinchalik istalgan odam foydalanishi ham mumkin. Bot har safar malum vaqtda sizning axborot tizimidagi parollaringizni yangilab turadi va sizga bu haqida xabar beradi. Shuningdek siz bu generatsiya amalini istalgan vaqtda amalga oshirishingizgiz mumkin. Endi bu botni ishlatish uchun sinov tariqasida sinov sayt yaratmiz. Bu keyinchalik axborot tizimi bilan almashtiriladi.

Saytning Front end qismi

React - Foydalanuvchi interfeyslarini yaratish uchun JavaScript kutubxonasi

Deklarativ

React interfaol foydalanuvchi interfeyslar(UI)ni qiyinchiliksiz yaratish imkoni beradi. Ilovangizdagi har bir holat uchun oddiy ko‘rinishni loyihalashtiring va React sizning ma’lumotlaringizni faqat o‘zgartirgan tarkibiy qismlarni belgilab ularni samarali ravishda yangilaydi[1].

Deklarativ ko‘rinishlar sizning kodingizni oldindan bashorat qilish va xatoliklarni onsonroq aniqlashga yordam beradi.

Komponentga-Asoslangan

O‘zining holatini boshqaradigan, keyin ularni murakkab UI ni yaratishda ishlatadigan enkapsulyatsiya qilingan komponentlar quring. Kopmonent logikasi shablonlarda emas balki JavaSciptda yozilgani sababli, ilovangizdagi boy ma’lumotlarni osongina uzatish va DOM ning holatini dahilsizligini ham saqlab qolasiz.

Bir marta o‘rganib, har yerda yozavering

Biz sizning boshqa texnologiya to‘plamingiz haqida har xil taxminlar qilmaymiz, shuning uchun mavjud kodni qayta yozmasdan React da yangi imkoniy(xususiyat)lar yartishishga imkon beradi.

Shuningdek React Node dan foydalangan holda server da va React Native dan foydalangan holda mobil ilovalarda ishlatiladi.

Oddiy komponent

React komponentlari render() metodidan foydalangan holda kirish ma’lumotini oladi va ularni ekranga chiqaradi. Quydagi misol JSX deb ataladigan XML ga o‘xshash sintaksisdan foydalanadi. Komponentga uzatilgan kirish ma’lumotlari render() tomonidan this.props orqali foydalanish imkonini beradi[2].

```
class HelloMessage extends React.Component {
  render() {
    return <div>Salom {this.props.name}</div>;
  }
}
root.render(<HelloMessage name="Alisher" />);
```

Holatli Komponent

Kirish ma’lumotlarini(`this.props` orqali kiradigan) olishdan tashqari, komponent ichki holat ma’lumotlari(`this.state` orqali kiradigan)ni ham boshqara oladi. Qachon komponentning holati(ya’ni state) o‘zgarganda, chizib bo‘lingan verstka `render()` ni qayta chaqirish orqali yangilanadi.

```
class Timer extends React.Component {
  constructor(props) {
    super(props);
```

```
      this.state = { seconds: 0 };
    }
    tick() {
      this.setState(state => ({
        seconds: state.seconds + 1
      }));
    }
    componentDidMount() {
      this.interval = setInterval(() => this.tick(),
1000);
    }
    componentWillUnmount() {
      clearInterval(this.interval);
    }
    render() {
      return (
        <div>
          Seconds: {this.state.seconds}
        </div>
      );
    }
  }
  root.render(<Timer />);
```

Ilova

`props` va `state` dan foydalanib, biz kichik Todo ilovasini yaratamiz. Ushbu misol mavjud ro‘yxatni hamda foydalanuvchi tomonidan kiritilgan matnlarni kuzatishda `state` dan foydalanadi. Event handler hodisalari chiziqli bo‘lishiga qaramssdan, aslida ular event delegation orqali to‘planadi va amalga oshiriladi[3].

```
  class TodoApp extends React.Component {
    constructor(props) {
      super(props);
      this.state = { items: [], text: '' };
      this.handleChange = this.handleChange.bind(this);
      this.handleSubmit = this.handleSubmit.bind(this);
    }
    render() {
```

```
      return (
        <div>
          <h3>TODO</h3>
          <TodoList items={this.state.items} />
          <form onSubmit={this.handleSubmit}>
            <label htmlFor="new-todo">
              Nima bajarilishi kerak?
            </label>
            <input
              id="new-todo"
              onChange={this.handleChange}
              value={this.state.text}
            />
            <button>
              Qo‘shish #{this.state.items.length + 1}
            </button>
          </form>
        </div>
      );
    }
    handleChange(e) {
      this.setState({ text: e.target.value });
    }
    handleSubmit(e) {
      e.preventDefault();
      if (this.state.text.length === 0) {
        return;
      }
      const newItem = {
        text: this.state.text,
        id: Date.now()
      };
      this.setState(state => ({
        items: state.items.concat(newItem),
        text: ''
      }));
    }
  }
  class TodoList extends React.Component {
```

```
    render() {
      return (
        <ul>
          {this.props.items.map(item => (
            <li key={item.id}>{item.text}</li>
          ))}
        </ul>
      );
    }
  }
  root.render(<TodoApp />);
```

Komponentda tashqi plagindan foydalish

React boshqa kutubxonalar va freymvorklar bilash ishlashga imkon beradi. Ushbu misolda `<textarea>` ni qiymatini doimiy o‘girish uchun **remarkable** tashqi Markdown kutubxonasidan foydalanilgan[4].

```
  class MarkdownEditor extends React.Component {
    constructor(props) {
      super(props);
      this.md = new Remarkable();
      this.handleChange = this.handleChange.bind(this);
      this.state = { value: 'Hello, **world**!' };
    }
    handleChange(e) {
      this.setState({ value: e.target.value });
    }
    getRawMarkup() {
      return { __html: this.md.render(this.state.value)
};
    }
    render() {
      return (
        <div className="MarkdownEditor">
          <h3>Input</h3>
          <label htmlFor="markdown-content">
            Markdown kiriting
          </label>
```

```
            <textarea
              id="markdown-content"
              onChange={this.handleChange}
              defaultValue={this.state.value}
            />
            <h3>Natija</h3>
            <div
              className="content"

dangerouslySetInnerHTML={this.getRawMarkup()}
            />
          </div>
        );
      }
    }
    root.render(<MarkdownEditor />);
```

Foydalanilgan adabiyotlar:

1. А.В. Васильков, И.А. Васильков. Безопасность и управление доступом в ИС. Учеб. пособие. М.:ФОРУМ:ИНФРА – МБ, 2019.- 368 с.
2. Р.А.Васильев. Управление информационной безопасностью. Курс лекций. Нижний Новгород. 2012.-169 с..
3. Stamp M. Information Security Principles and Practice 2nd Edition – 2011. Canada.– 608 p.
4. Islomov S. Z. et al. New authentication scheme for cloud computing //Journal of Advanced Research in Dynamical and Control Systems. – 2018. – Т. 10. – №. 10. – С. 2316-2319.