МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«СЕВАСТОПОЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Институт Информационных технологий и управления в технических системах

Кафедра Информационные системы

|  |  |  |  |
| --- | --- | --- | --- |
| № | Дата поступления на кафедру | Подпись отв. за регистрацию | Подпись преподавателя |
|  |  |  |  |
|  |  |  |  |

**ОТЧЕТ**

о производственной (по получению профессиональных умений и опыта профессиональной деятельности) практике

в \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ИП Коркина\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(наименование организации)

Выполнил \_\_\_Долженко Илья\_\_\_\_\_\_

\_Андреевич\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Фамилия И.О. обучающегося)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ИС/б-17-2-о\_\_\_\_\_\_\_\_\_\_\_\_\_

(шифр группы)

Направление / специальность 09.03.02

Информационные системы и технологии

(код, наименование)

Руководитель практики от Университета \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(должность)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_Заикина Е.Н.\_\_\_\_\_\_\_\_\_\_\_\_

(Фамилия И.О. руководителя)

Севастополь

2020 г.

В ходе прохождения практики были повторены и закреплены знания языка программирования Java Script и библиотеки ReactJS, изучена библиотека MobX для управления состоянием приложения и освоен ReactUI-фреймворк Material UI.

React — JavaScript-библиотека с открытым исходным кодом для разработки пользовательских интерфейсов. React разрабатывается и поддерживается Facebook, Instagram и сообществом отдельных разработчиков и корпораций. React может использоваться для разработки одностраничных и мобильных приложений. Его цель — предоставить высокую скорость, простоту и масштабируемость. В качестве библиотеки для разработки пользовательских интерфейсов React часто используется с другими библиотеками, такими как MobX, Redux и GraphQL.

MobX — это автономная библиотека, для управления фронтенд-состоянием приложения. MobX обеспечивает консистентность и согласованность внутреннего состояния фронтенд-приложения, предоставляя удобные инструменты для его изменения.

Material UI — это набор компонентов React, которые реализуют Google Material Design. Эти компоненты работают изолированно, это означает, что они являются самоподдерживающимися и вводят только те стили, которые они должны отображать.

Первой задачей на проекте являлась задача создания форм авторизации и регистрации. Формы должны содержать валидацию и быть адаптивными. Для реализации отправки данных с этих форм на сервер была использована библиотека axios, которая позволяет быстро и легко отправить любой HTTP-запрос на сторону сервера.

Код валидации форм из файла LoginStore.js:

@action

    validateForm = () => {

*this*.error = "";

*for* (let key in *this*.registrationForm) {

*if* (*this*.registrationForm[key] === "") {

*this*.error = "Поля не должны быть пустыми";

*return* false;

            }

        }

*for* (let key in *this*.formErrors) {

*if* (Boolean(*this*.formErrors[key])) *return* false;

        }

*return* true;

    };

Конструктор класса, в котором описаны правила валидации:

    constructor() {

        reaction(

            () => *this*.registrationForm.phone,

            phone => phone && (*this*.formErrors.phone = phoneNumber(phone))

        );

        reaction(

            () => *this*.registrationForm.username,

            username => username && (*this*.formErrors.username = minLengthField(username, 3))

        );

        reaction(

            () => *this*.registrationForm.email,

            email => email && (*this*.formErrors.email = emailAddress(email))

        );

        reaction(

            () => *this*.registrationForm.password,

            password => password && (*this*.formErrors.password = minLengthField(password, 5))

        );

    }

Метод отправки данных на сервер:

    @action

    doRegistration = () => {

*if* (!*this*.validateForm()) *return*;

*this*.pending = true;

        let signUpformData = new FormData();

*for* (let key in *this*.registrationForm) {

            signUpformData.append(key, *this*.registrationForm[key]);

        }

        axiosInstance({

            method: "post",

            url: "/api/v1/user/sign-up",

            data: signUpformData,

        })

            .then(response => {

*if* (response.data.error === 0) {

*this*.redirect = true;

                } *else* {

*this*.error = "Данные имя пользователя или email уже заняты";

                }

            })

            .catch(error => {})

            .finally(() => (*this*.pending = false));

    };

Функция для выполнения авторизации, содержит код отправки запроса на сервер и код запоминания пользователя, если он поставил галочку “запомнить меня”:

@action

    doLogin = isRemember => {

*if* (!*this*.validateForm()) *return*;

*this*.pending = true;

        let signInformData = new FormData();

*for* (let key in *this*.loginForm) {

            signInformData.append(key, *this*.loginForm[key]);

        }

        axiosInstance({

            method: "post",

            url: "/api/v1/user/sign-in",

            data: signInformData,

        })

            .then(response => {

*if* (response.data.error === 0) {

                    isRemember ? localStorage.setItem("accessToken", response.data.token) : sessionStorage.setItem("accessToken", response.data.token);

*this*.authorizationStore.fetchCurrentUser();

                } *else* {

*this*.error = "Неверный логин или пароль";

                }

            })

            .catch(error => {})

            .finally(() => (*this*.pending = false));

    };

Файл AuthForm.js который реализует выше описанные функции:

const useStyles = makeStyles(theme => ({

    authorizationTitle: {

        marginTop: "10px",

    },

    centerLoader: {

        display: "flex",

        alignItems: "center",

        justifyContent: "center",

        height: "100%",

        width: "100%",

    },

}));

*export* const \_AuthForm = ({

    loginForm,

    setFormValue,

    doLogin,

    pending,

    formErrors,

    validateForm,

    error,

}) => {

    const classes = useStyles();

    const [isRemember, setIsRemember] = useState(false);

*return* (

        <Container *maxWidth*="xs">

            <div *className*={classes.authorization}>

                <Typography *variant*="h5" *className*={classes.authorizationTitle}>

                    Вход

                </Typography>

                <Grid *container* *spacing*={2}>

                    <Grid *item* *xs*={12}>

                        <TextField

*label*="Логин"

*value*={loginForm.username}

*onChange*={event => {

                                setFormValue("username", event.target.value);

                            }}

*error*={Boolean(formErrors.username)}

*helperText*={formErrors.username}

*fullWidth*

                        />

                    </Grid>

                    <Grid *item* *xs*={12}>

                        <TextField

*label*="Пароль"

*type*="password"

*value*={loginForm.password}

*onChange*={event => {

                                setFormValue("password", event.target.value);

                            }}

*error*={Boolean(formErrors.password)}

*helperText*={formErrors.password}

*fullWidth*

                        />

                    </Grid>

                    <Grid *item* *xs*={12}>

                        <FormControlLabel

*control*={

                                <Checkbox

*checked*={isRemember}

*onChange*={event => {

                                        setIsRemember(!isRemember);

                                    }}

*color*="primary"

                                />

                            }

*label*="Запомнить меня"

                        />

                    </Grid>

                    <Grid *item* *xs*={12} *className*={classes.centerLoader}>

                        <Loader *loading*={pending} />

                    </Grid>

                    <Grid *item* *xs*={12}>

                        <Button

*variant*="contained"

*color*="primary"

*onClick*={() => {

                                doLogin(isRemember);

                            }}

*fullWidth*

                        >

                            Войти

                        </Button>

                    </Grid>

                    <Grid *item* *xs*={12}>

                        <Typography *variant*="body2" *color*="error">

                            {error}

                        </Typography>

                    </Grid>

                    <Grid *item* *xs*={12}>

                        <Link *view*={Routes.register} *store*={routerStore}>

                            <Typography *variant*="body2">

                                Нет аккаунта? Зарегистрируйтесь

                            </Typography>

                        </Link>

                    </Grid>

                </Grid>

            </div>

        </Container>

    );

};

const mapMoxToProps = ({ login }) => ({

    loginForm: login.loginForm,

    setFormValue: login.setFormValue,

    doLogin: login.doLogin,

    pending: login.pending,

    formErrors: login.formErrors,

    validateForm: login.validateForm,

    error: login.error,

});

*export* const AuthForm = inject(mapMoxToProps)(observer(\_AuthForm));

На рисунке 1 представлена форма входа на сайт, а на рисунке 2 форма регистрации с результатами валидации.

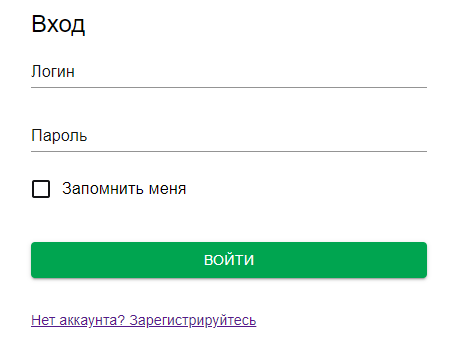


Рисунок 1 – Форма входа на сайт

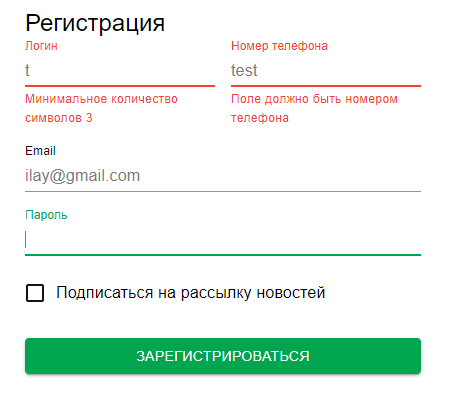


Рисунок 2 – Форма регистрации с результатами валидации

Для создания авторизации через социальные сети была выбрана библиотека React Social Login, которая позволяет входить на сайт через такие социальные сети как: ВК, Facebook, Google и Twitter.

При создании страницы личного кабинета, сначала была реализована визуальная составляющая, представленная на рисунке 3. После чего установлена связь с сервером для заполнения страницы данными о пользователе.



Рисунок 3 – Личный кабинет пользователя

В компоненте ProfilePage содержится основная сетка для построения структуры страницы. Данная сетка позволяет разбивать целую страницу на подкомпоненты, что позволяет лучше структурировать код и перерисовывать только тот компонент, который изменяется. Содержимое данного файла представлено ниже:

*export* const ProfilePage = () => {

    const classes = useStyles();

*return* (

        <>

            <div *className*={classes.root}>

                <Grid *container*>

                    <Grid *item* *lg*={3} *md*={3}>

                        <Navigation />

                    </Grid>

                    <Grid *className*={classes.profileBody} *item* *lg*={7} *md*={12}>

                        <Grid *container* *spacing*={4}>

                            <Grid *item* *lg*={4} *md*={4} *sm*={12} *xs*={12}>

                                <UserInfo />

                            </Grid>

                            <Grid *item* *lg*={4} *md*={4} *sm*={12} *xs*={12}>

                                <Balance />

                            </Grid>

                            <Grid *item* *lg*={4} *md*={4} *sm*={12} *xs*={12}>

                                <Exchange />

                            </Grid>

                        </Grid>

                        <Grid *container* *spacing*={4}>

                            <Grid *item* *lg*={12} *md*={12}>

                                <UserRating />

                            </Grid>

                        </Grid>

                    </Grid>

                    <Grid  *className*={classes.profileMessages} *item* *lg*={2}>

                        <Messages />

                    </Grid>

                </Grid>

            </div>

        </>

    );

};

Ниже представлен код навигационного меню из файла Navigation.js:

*export* const Navigation = () => {

*return* (

        <nav *className*="navigation">

            <div *className*="navigation-head">

                <div *className*="logo">

                    <Link *view*={Routes.home} *store*={routerStore}>

                        <img *src*={logo} *alt*="Golden ticket" />

                    </Link>

                </div>

                <div *className*="profile-name">

                    <div *className*="profile-name-logo">

                        <div *className*="user-logo">

                            <img *src*={userImage} *alt*="" />

                        </div>

                        <ul *className*="user-info">

                            <li *className*="user-info-name">Users name</li>

                            <li *className*="user-info-balance gold">100.25 P</li>

                        </ul>

                    </div>

                    <div *className*="profile-name-btn">

                        <Button *className*="primary-button">Пополнить баланс</Button>

                    </div>

                </div>

                <div *className*="profile-rating">

                    <div *className*="rating-stars">

                        <img *src*={star} *alt*="" />

                        <img *src*={star} *alt*="" />

                        <img *src*={star2} *alt*="" />

                        <img *src*={star2} *alt*="" />

                        <img *src*={star2} *alt*="" />

                    </div>

                    <div *className*="rating-value">

                        <div *className*="value-name">

                            <span *className*="gold">Рейтинг:</span> <p>Серебряный</p>

                        </div>

                        <div *className*="value-load">

                            <img *src*={load} *alt*="" />

                        </div>

                        <div *className*="value-number">

                            <p>

                                <span>N</span> билетов до рейтинга Золотой

                            </p>

                        </div>

                    </div>

                </div>

                <Line />

            </div>

            <div *className*="navigation-menu">

                <Link *view*={Routes.home} *store*={routerStore}>

                    <Icon>house</Icon>Главная

                </Link>

                <Link *view*={Routes.home} *store*={routerStore}>

                    <Icon>person</Icon>Профиль

                </Link>

                <Link *view*={Routes.home} *store*={routerStore}>

                    <Icon>casino</Icon>Игра

                </Link>

                <Link *view*={Routes.home} *store*={routerStore}>

                    <Icon>article</Icon>Правила игры

                </Link>

                <Link *view*={Routes.home} *store*={routerStore}>

                    <Icon>credit\_card</Icon>Магазин

                </Link>

                <Link *view*={Routes.home} *store*={routerStore}>

                    <Icon>insert\_chart</Icon>Рейтинг игроков

                </Link>

                <Link *view*={Routes.home} *store*={routerStore}>

                    <Icon>settings</Icon>Настройки

                </Link>

                <Line />

            </div>

            <div *className*="navigation-exit">

                <Button *className*={"exit-btn"}>Выход</Button>

            </div>

        </nav>

    );

};

Компонент просмотра баланса пользователя:

const \_Balance = ({ balance, pending }) => {

*return* (

        <>

            <div *className*="user-currency-card">

                <h1 *className*="user-info-card\_\_title">Наш баланс</h1>

                <div *className*="line"></div>

                <Grid

*container*

*justify*="center"

*className*="user-currency-card-inner"

                >

                    <Grid *item*>

                        <div *className*="currency">

                            <div *className*="currency-icon">

                                <WalletIcon *width*={40} *height*={35} />

                            </div>

                            <div *className*="currency\_\_info">

                                <h1 *className*="currency\_\_title">Деньги</h1>

                                <span *className*="currency\_\_balance">

                                    <Loader *size*={10} *loading*={pending} />

                                    {balance.money} P

                                </span>

                            </div>

                        </div>

                    </Grid>

                    <div *className*="line"></div>

                    <Grid *item*>

                        <div *className*="currency">

                            <div *className*="currency-icon">

                                <CoinsIcon *width*={40} *height*={35} />

                            </div>

                            <div *className*="currency\_\_info">

                                <h1 *className*="currency\_\_title">Монеты</h1>

                                <span *className*="currency\_\_balance">

                                    <Loader *size*={10} *loading*={pending} />

                                    {balance.coins} монет

                                </span>

                            </div>

                        </div>

                    </Grid>

                    <div *className*="line"></div>

                    <Grid *item*>

                        <div *className*="currency">

                            <div *className*="currency-icon">

                                <CouponIcon *width*={40} *height*={35} />

                            </div>

                            <div *className*="currency\_\_info">

                                <h1 *className*="currency\_\_title">Купоны</h1>

                                <span *className*="currency\_\_balance">

                                    <Loader *size*={10} *loading*={pending} />

                                    {balance.tickets} купон

                                </span>

                            </div>

                        </div>

                    </Grid>

                    <div *className*="line"></div>

                </Grid>

                <div *className*="user-info-card\_\_primary-button">

                    <button *className*="primary-button">Пополнить баланс</button>

                </div>

            </div>

        </>

    );

};

const mapMoxToProps = ({ profile }) => ({

    balance: profile.balance,

    pending: profile.pending,

});

*export* const Balance = inject(mapMoxToProps)(

    observer(\_Balance)

);

При выполнении основной задачи, связанной с игровым процессом, была проделана большая работа, так как требовалось понимание схемы основного процесса. Для начала было реализовано игровое поле, затем продуман интерфейс выбора необходимых комбинаций чисел. Интерфейс страницы с игровым полем представлен на рисунке 4.

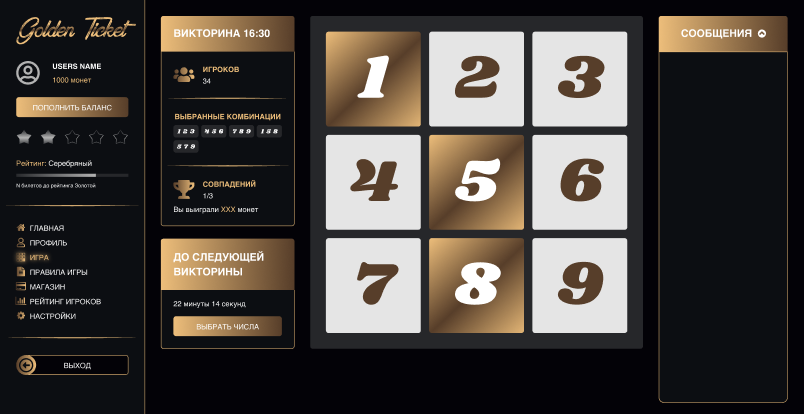


Рисунок 4 – Страница проведения игры

Код компонента, отвечающего за отображение основной информации о игре, ее игроков, количества совпадений и кнопку выбора чисел представлен ниже:

const \_Quiz = ({ currentUser }) => {

    const classes = useStyles();

*return* (

        <>

            <div *className*="quiz-info-card">

                <h1 *className*="quiz-info-card\_\_title">Викторина</h1>

                <div *className*="line"></div>

                <div *className*="user-icon">

                    <QuizIcon *width*={138} *height*={138} />

                </div>

                <div *className*="line"></div>

                <div *className*="quiz-info-card\_\_text-input">

                    <OutlinedInput

*className*={classes.textInput}

*endAdornment*={

                            <InputAdornment *position*="end">

                                <PenIcon *width*={16} *height*={18} />

                            </InputAdornment>

                        }

*aria-describedby*="outlined-weight-helper-text"

*inputProps*={{

                            "aria-label": "weight",

                        }}

*labelWidth*={0}

                    />

                </div>

                <div *className*="line"></div>

                <div *className*="user-info-card\_\_primary-button">

                    <button *className*="primary-button">Выбрать числа</button>

                </div>

            </div>

        </>

    );

};

const mapMoxToProps = ({ quiz }) => ({

    numbers: quiz.numbers

});

*export* const Quiz = inject(mapMoxToProps)(observer(\_Quiz));

Также начат компонент Chat для отображения пользовательского чата:

const useStyles = makeStyles(theme => ({

    header: {

        background: "linear-gradient(270deg, #573E2A 0%, #EDBE7B 100%);",

        border: "1px solid #EDBE7B",

        borderRadius: "5px 5px 0 0",

        fontWeight: "bold",

        fontSize: 20,

        height: 68,

        padding: "0 40px 0 40px",

    },

    body: {

        backgroundColor: "#0C0E12",

        border: "1px solid #EDBE7B",

        borderRadius: "0 0 5px 5px",

        height: 660,

    }

}));

*export* const Messages = () => {

    const classes = useStyles();

*return* (

        <>

            <div *className*="messages">

                <MessagesAccordion>

                    <AccordionSummary

*className*={classes.header}

*expandIcon*={<ArrowDropDownCircleRoundedIcon />}

*aria-controls*="panel1a-content">

                        <Typography *variant*="inherit">Сообщения</Typography>

                    </AccordionSummary>

                    <AccordionDetails *className*={classes.body}>

                    </AccordionDetails>

                </MessagesAccordion>

            </div>

        </>

    );

};

ВЫВОД

В ходе прохождения практики был получен опыт работы в команде над настоящим проектом. Знания, которые были приобретены во время обучения на дисциплине «WEB Технологии» оказались неоценимо полезны, ведь за время обучения были рассмотрены основы создания сайтов, язык гипертекстовой разметки HTML, каскадная таблица стилей CSS и язык программирования Java Script. Также в процессе работы над проектом были укреплены знания библиотеки React JS и был освоен ReactUI-фреймворк Material UI. Были получены знания работы с библиотекой управления состоянием приложения MobX и опыт совместного написания кода с помощью сервиса GitHub.