Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №3

По дисциплине: «ВТ»

Тема: “Защита данных”

Выполнил:

Студент 3 курса

Группы ПО-7

Комиссаров А.Е.

Проверил:

Михняев А.Л.

2022

**Цель работы:** Познакомиться с подходами защиты данных в веб- приложениях.

Сгенерировать HTTP-запрос осуществляющий **шифрованную** передачу данных в гиперссылке: **параметры** строки информации передавать в хешированной форме, данные – в шифрованной форме. Осуществить дешифрацию полученной информации и вывод ее на экран.

**Ход работы.**

1. Форма и логика отправки зашифрованных данных.

const Form = () => {

const {

register,

handleSubmit,

reset,

formState: { errors },

} = useForm<IFormFields>({

resolver: yupResolver(schema),

defaultValues: {

name: "Andrey",

position: "Lead",

department: "Development",

email: "test123@gmail.com",

phone: "+375299792099",

comment: "Lorem ipsum dolor",

},

})

const navigate = useNavigate()

const onSubmit: SubmitHandler<IFormFields> = async data => {

reset()

const cipherText = CryptoJS.AES.encrypt(JSON.stringify(data), KEY).toString()

const response = await axios.post("http://localhost:8001/api/crypto/encrypt", { data: cipherText })

navigate("/results", {

state: {

requestedData: data,

encryptedData: cipherText,

decryptedData: response.data,

},

})

}

return (

<>

<form onSubmit={handleSubmit(onSubmit)} noValidate className="w-full max-w-sm">

<h2 className="mb-6 text-center text-3xl font-bold tracking-wider">My Form</h2>

<TextInput {...register("name")} label="Name" error={errors.name?.message} />

<TextInput {...register("position")} label="Position" error={errors.position?.message} />

<TextInput {...register("department")} label="Department" error={errors.department?.message} />

<TextInput {...register("phone")} label="Phone" error={errors.phone?.message} />

<TextInput {...register("email")} label="Email" error={errors.email?.message} />

<TextInput {...register("comment")} label="Comment" error={errors.comment?.message} />

<button type="submit" className="btn-primary btn mt-3 w-full text-lg">

Send

</button>

</form>

</>

)

}

1. Дешифрование данные на сервере:

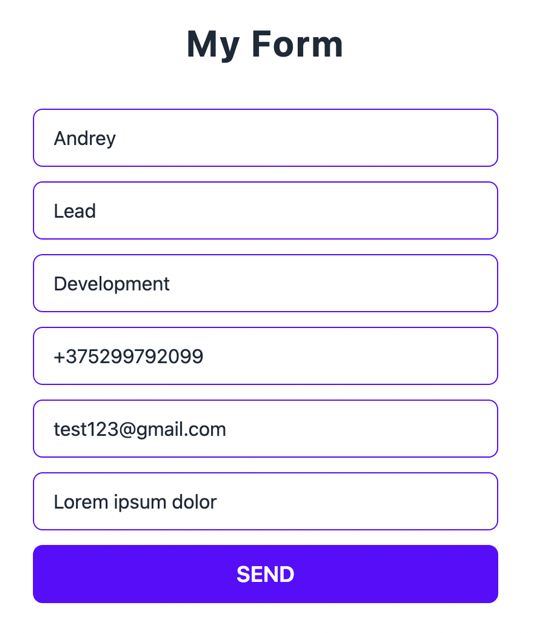
async encryptData(data: { data: string }): Promise<Person> {

var bytes = CryptoJS.AES.decrypt(data.data, "SECRET")

var decryptedData = JSON.parse(bytes.toString(CryptoJS.enc.Utf8))

return decryptedData

}



2. Разработать функционал, осуществляющий:  
a. регистрацию и аутентификацию пользователей  
b. проверку уникальности учетной записи (логина) при регистрации c. хранение пароля в базе данных в шифрованной форме

3. Форма регистрации и авторизации.

const AuthForm = () => {

const {

register: registerField,

reset,

handleSubmit,

formState: { errors },

} = useForm<IAuthFields>({

resolver: yupResolver(schema),

})

const [error, setError] = useState("")

const navigate = useNavigate()

const setToLS = (data: { user: IAuthFields; accessToken: string }) => {

localStorage.setItem("auth", JSON.stringify(data))

}

const register: SubmitHandler<IAuthFields> = async data => {

try {

const response = await axios.post("http://localhost:8001/api/auth/register", data)

setToLS(response.data)

navigate("/")

reset()

} catch (error) {

setError((error as any).response.data.message)

}

}

const login: SubmitHandler<IAuthFields> = async data => {

try {

const response = await axios.post("http://localhost:8001/api/auth/login", data)

setToLS(response.data)

navigate("/")

reset()

} catch (error) {

setError((error as any).response.data.message)

}

}

return (

<form className="w-full max-w-sm" noValidate>

<h2 className="text-center text-2xl font-bold text-white">Auth</h2>

<TextInput label="Email" {...registerField("email")} error={errors.email?.message} />

<TextInput label="Password" {...registerField("password")} error={errors.password?.message} type="password" />

{error && <span className="mt-3 text-sm font-light text-red-600">{error}</span>}

<button type="button" className="btn btn-primary mt-3 w-full text-lg" onClick={handleSubmit(register)}>

Register

</button>

<button type="button" className="btn btn-primary mt-3 w-full text-lg" onClick={handleSubmit(login)}>

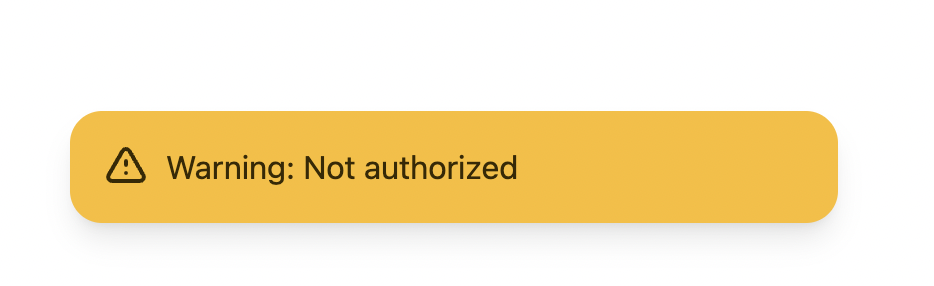
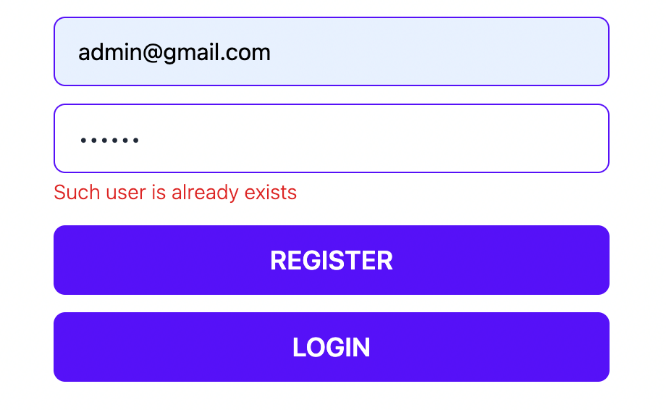
Login

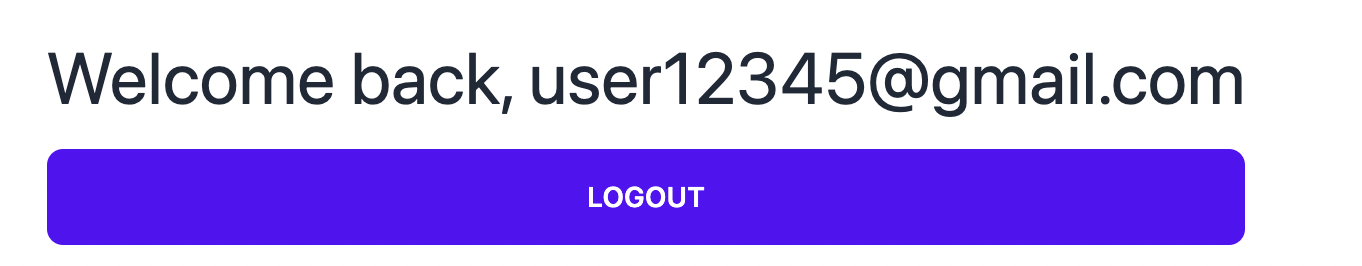
</button>

</form>

)

}





1. Регистрация и авторизация на сервере:

async register(dto: AuthDto): Promise<AuthResponse> {

const candidate = await this.prismaService.user.findUnique({

where: { email: dto.email },

})

if (candidate) {

throw new BadRequestException("Such user is already exists")

}

const salt = await genSalt(5)

const hashedPassword = await hash(dto.password, salt)

const user = await this.prismaService.user.create({

data: {

email: dto.email,

password: hashedPassword,

},

})

const accessToken = await this.jwtService.sign({ id: user.id })

return {

user,

accessToken,

}

}

async login(dto: AuthDto): Promise<AuthResponse> {

const user = await this.validateUser(dto)

const accessToken = await this.jwtService.sign({ id: user.id })

return {

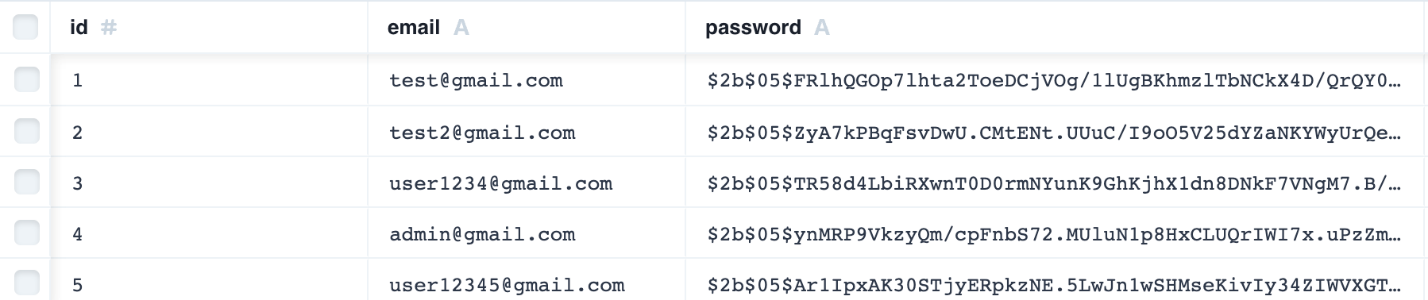
user,

accessToken,

}

}

1. Пароли в базе данных хранятся в захешированном виде.



**Вывод:** я познакомился с подходами защиты данных в веб- приложениях.