

Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное образовательное учреждение высшего образования
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчет

по лабораторной работе №6
по дисциплине «Web-программирование»

Автор: Шарапков Егор Викторович М33081

Преподаватель: Приискалов Роман Андреевич



УНИВЕРСИТЕТ ИТМО

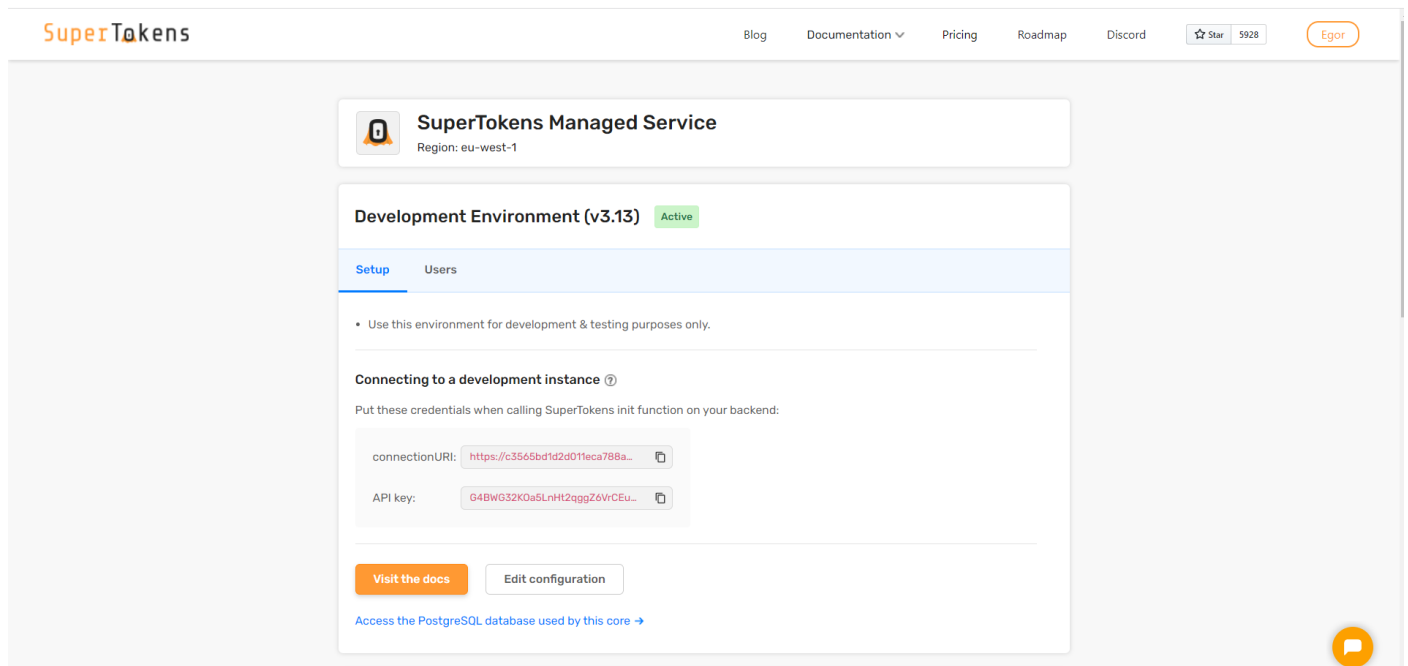
Санкт-Петербург 2022

Добавление авторизации и пользовательских сессий.

Для удовлетворения всех стандартов безопасности, которые существуют на сегодняшний день, предлагается вместо того, чтобы разрабатывать свой механизм авторизации, воспользоваться поставщиками таких услуг как Authorization as a Service.

Я выбрал сервис Supertokens.

Регистрируюсь на сайте.



Добавляем эти значения в переменные окружения (локально и в хероку).

Устанавливаем supertokens

```
npm i -s supertokens-node
```

Далее создаем модуль для аутентификации и авторизации.

```
nest g module auth
```

```
1 import {
2   MiddlewareConsumer,
3   Module,
4   NestModule,
5   DynamicModule,
6 } from '@nestjs/common';
7
8 import { AuthMiddleware } from './auth.middleware';
9 import { ConfigInjectionToken, AuthModuleConfig } from './config.interface';
10 import { SupertokensService } from './supertokens/supertokens.service';
11
12 @Module({ metadata: {
13   providers: [SupertokensService],
14   exports: [],
15   controllers: [],
16 }})
17 export class AuthModule implements NestModule {
18   configure(consumer: MiddlewareConsumer) {
19     consumer.apply(AuthMiddleware).forRoutes('*');
20   }
21
22   static forRoot({ connectionURI, apiKey, appInfo }: AuthModuleConfig): DynamicModule {
23     return {
24       providers: [
```

добавляем интерфейс конфигурации в папку auth, которая была создана для модуля

```
1 import { AppInfo } from "supertokens-node/lib/build/types";
2
3 export const ConfigInjectionToken = "ConfigInjectionToken";
4
5 export type AuthModuleConfig = {
6   appInfo: AppInfo;
7   connectionURI: string;
8   apiKey?: string;
9 }
```

Обновим модуль приложения, чтобы использовать динамический модуль.

```
4 import { ConfigModule } from '@nestjs/config';
5 import { DeviceModule } from '../device/device.module';
6 import { BasketModule } from '../basket/basket.module';
7 import { AuthModule } from '../auth/auth.module';
8
9 @Module({ metadata: {
10   imports: [ConfigModule.forRoot(), AuthModule.forRoot({connectionURI, apiKey, appInfo}: {
11     connectionURI: process.env.ConnectionURI,
12     apiKey: process.env.SecretApiKey,
13     appInfo: {
14       appName: "applemarket",
15       apiDomain: "http://localhost:3000",
16       websiteDomain: "https://applemarketru.herokuapp.com",
17       apiBasePath: "/auth",
18       websiteBasePath: "/auth",
19     }
20   })), UserModule, DeviceModule, BasketModule],
21   controllers: [AppController],
22   providers: [],
23 })
24 export class AppModule {}
```

Создадим сервис

```
nest g service supertokens auth
```

```
1 import { Inject, Injectable } from '@nestjs/common';
2 import supertokens from "supertokens-node";
3 import Session from 'supertokens-node/recipe/session';
4 import EmailPassword from 'supertokens-node/recipe/emailpassword';
5
6 import { ConfigInjectionToken, AuthModuleConfig } from '../config.interface';
7
8 @Injectable()
9 export class SupertokensService {
10   constructor(@Inject(ConfigInjectionToken) private config: AuthModuleConfig) {
11     supertokens.init({ config: {
12       appInfo: config.appInfo,
13       supertokens: {
14         connectionURI: config.connectionURI,
15         apiKey: config.apiKey,
16       },
17       recipeList: [
18         EmailPassword.init(),
19         Session.init({ config: {
20           cookieSameSite: "strict"
21         } }),
22       ],
23     } })
24   }
25 }
```

Предоставление API-интерфейсов SuperTokens с помощью промежуточного программного обеспечения

```
nest g middleware auth auth
```

```
1 import { Injectable, NestMiddleware } from '@nestjs/common';
2 import { middleware } from 'supertokens-node/framework/express';
3
4 @Injectable()
5 export class AuthMiddleware implements NestMiddleware {
6   supertokensMiddleware: any;
7
8   constructor() {
9     this.supertokensMiddleware = middleware();
10  }
11
12  use(req: Request, res: any, next: () => void) {
13    return this.supertokensMiddleware(req, res, next);
14  }
15 }
```

Далее необходимо обновить параметры политики CORS. В файл main.ts нам нужно добавить следующее:

```
app.enableCors({ options: {
  origin: ['https://applemarkettru.herokuapp.com'],
  allowedHeaders: ['content-type', ...supertokens.getAllCORSHeaders()],
  credentials: true,
}});
```

Добавляем фильтр исключений.

```
nest g filter auth auth
```

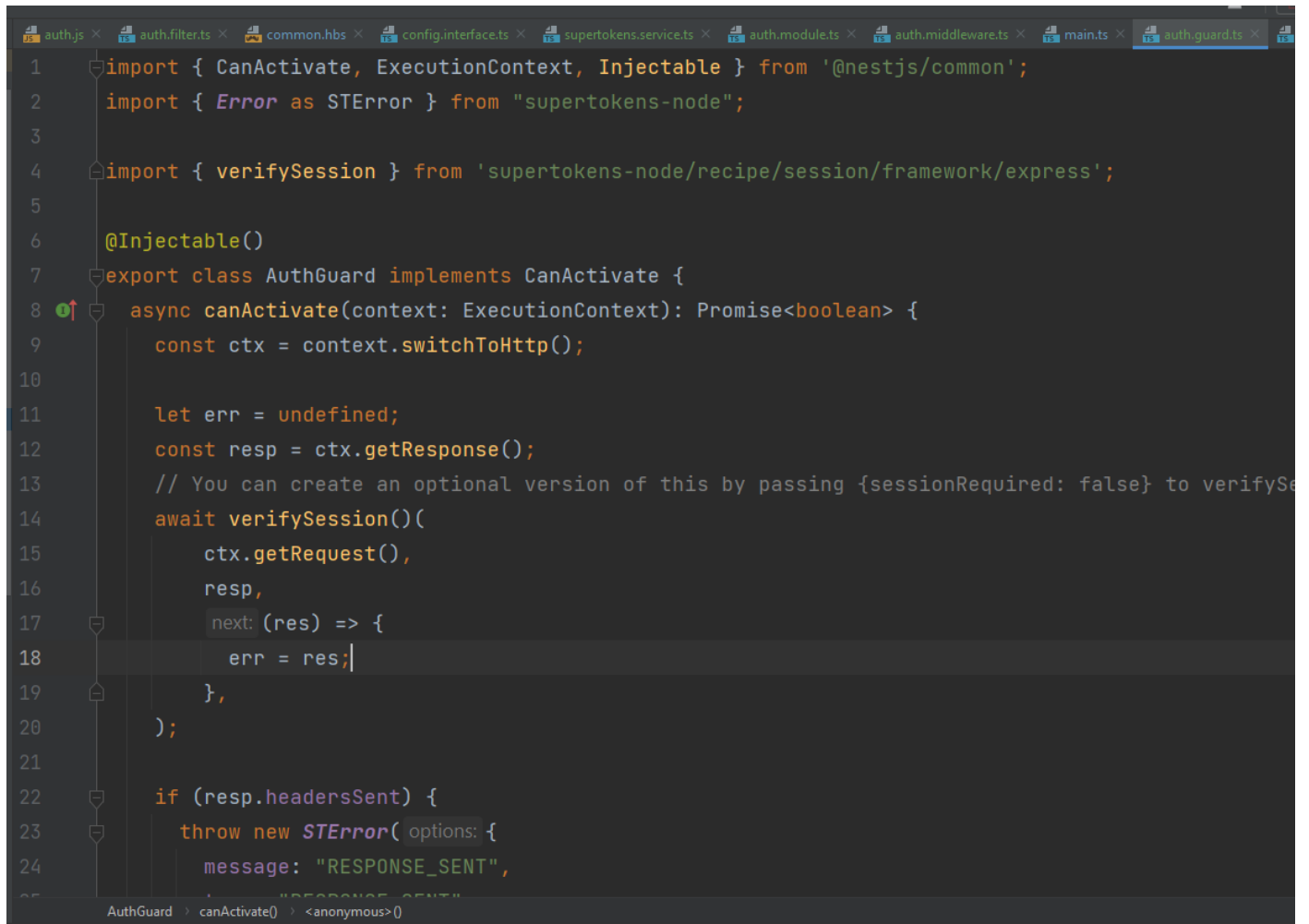
```
1 import { ExceptionFilter, Catch, ArgumentsHost } from '@nestjs/common';
2 import { Request, Response, NextFunction, ErrorRequestHandler } from 'express';
3
4 import { errorHandler } from 'supertokens-node/framework/express';
5 import { Error as STError } from 'supertokens-node';
6
7 @Catch(STError)
8 export class SupertokensExceptionHandler implements ExceptionFilter {
9   handler: ErrorRequestHandler;
10
11   constructor() {
12     this.handler = errorHandler();
13   }
14
15   catch(exception: Error, host: ArgumentsHost) {
16     const ctx = host.switchToHttp();
17
18     const resp = ctx.getResponse<Response>();
19     if (resp.headersSent) {
20       return;
21     }
22
23     this.handler(
24       exception,
```

Регистрируем фильтр.

```
app.useGlobalFilters(new SupertokensExceptionFilter());
```

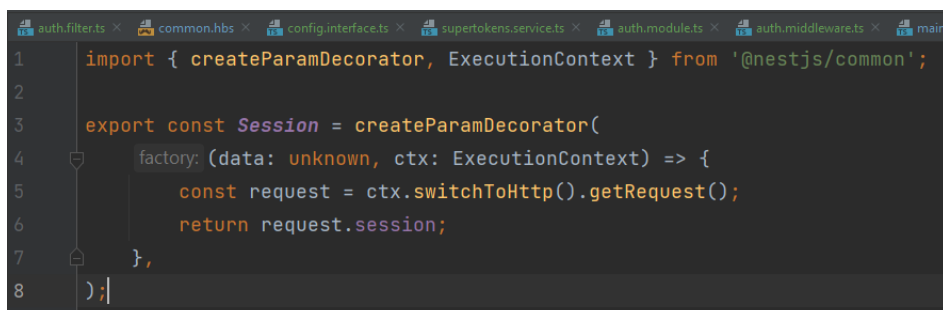
Теперь, когда библиотека настроена, можно добавить guard для защиты API

```
nest g guard auth auth
```



```
1 import { CanActivate, ExecutionContext, Injectable } from '@nestjs/common';
2 import { Error as STError } from "supertokens-node";
3
4 import { verifySession } from 'supertokens-node/recipe/session/framework/express';
5
6 @Injectable()
7 export class AuthGuard implements CanActivate {
8   async canActivate(context: ExecutionContext): Promise<boolean> {
9     const ctx = context.switchToHttp();
10
11     let err = undefined;
12     const resp = ctx.getResponse();
13     // You can create an optional version of this by passing {sessionRequired: false} to verifySession
14     await verifySession()(
15       ctx.getRequest(),
16       resp,
17       next: (res) => {
18         err = res;
19       },
20     );
21
22     if (resp.headersSent) {
23       throw new STError( options: {
24         message: "RESPONSE_SENT",
25         "RESPONSE_SENT"
```

Теперь можно добавить декоратор параметров для доступа к уже проверенному сеансу в моих API.



```
1 import { createParamDecorator, ExecutionContext } from '@nestjs/common';
2
3 export const Session = createParamDecorator(
4   factory: (data: unknown, ctx: ExecutionContext) => {
5     const request = ctx.switchToHttp().getRequest();
6     return request.session;
7   },
8 );
```

Теперь можно добавить защищенный метод к контроллеру, который получает проверенный сеанс в качестве параметра.

```
@ApiOperation( options: {
  summary: 'Get all devices'
})
@UseGuards(AuthGuard)
@ApiBasicAuth()
@Get()
async getDevices(@Session() session: SessionContainer): Promise<Device[]> {
  return await this.deviceService.getDevices();
}
```

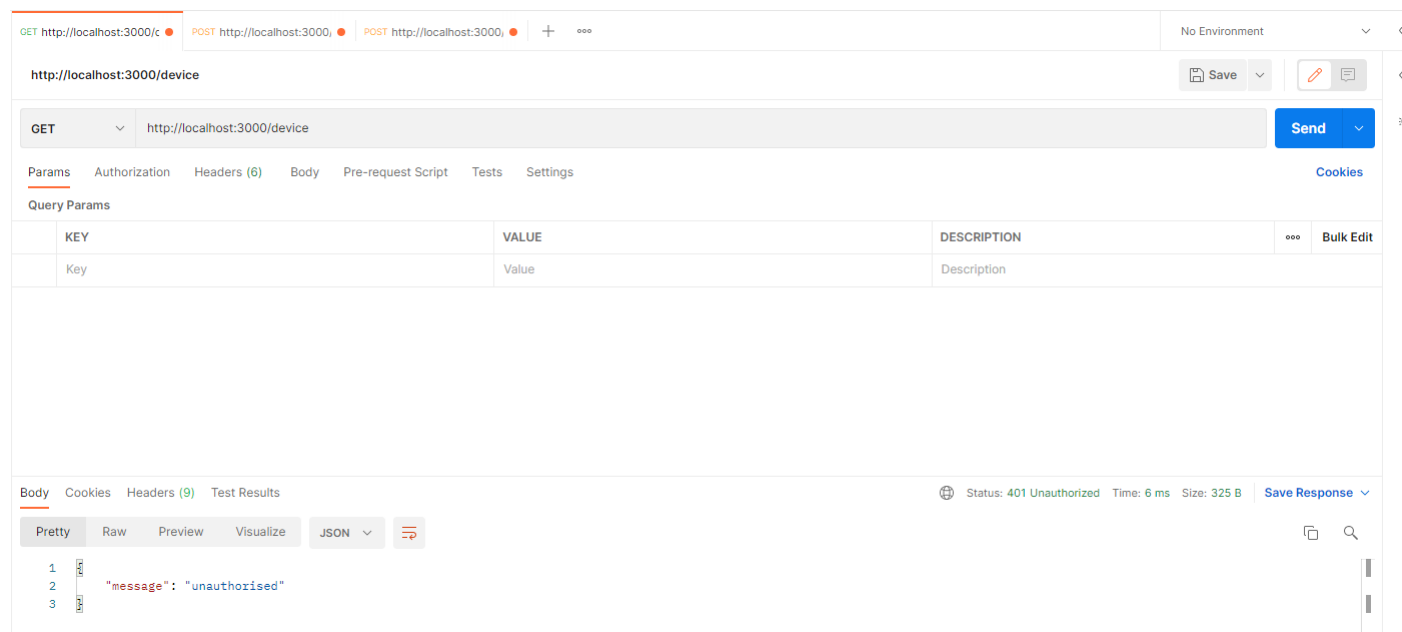
```
@ApiOperation( options: {
  summary: 'Delete a device by id'
})
@ApiResponse( options: {
  status: 400,
  description: 'Invalid id format',
})
@UseGuards(AuthGuard)
@ApiBasicAuth()
@Delete( path: ':{id}')
async delete(@Param( property: 'id', ParseIntPipe) id: number, @Session() session: SessionContainer): Promise<void> {
  const userId = session.getUserId();
  console.log(userId);
  return await this.deviceService.deleteDevice(id);
}
```

При добавлении защиты авторизации мы можем проверить swagger, и он покажет знак блокировки рядом с конечной точкой:

device		^
GET	/device Get all devices	✓ 🔒
POST	/device Create a device by title, description and price	✓
GET	/device/{id} Get a device by id	✓
DELETE	/device/{id} Delete a device by id	✓ 🔒
POST	/device/{id}/update Update device by id	✓

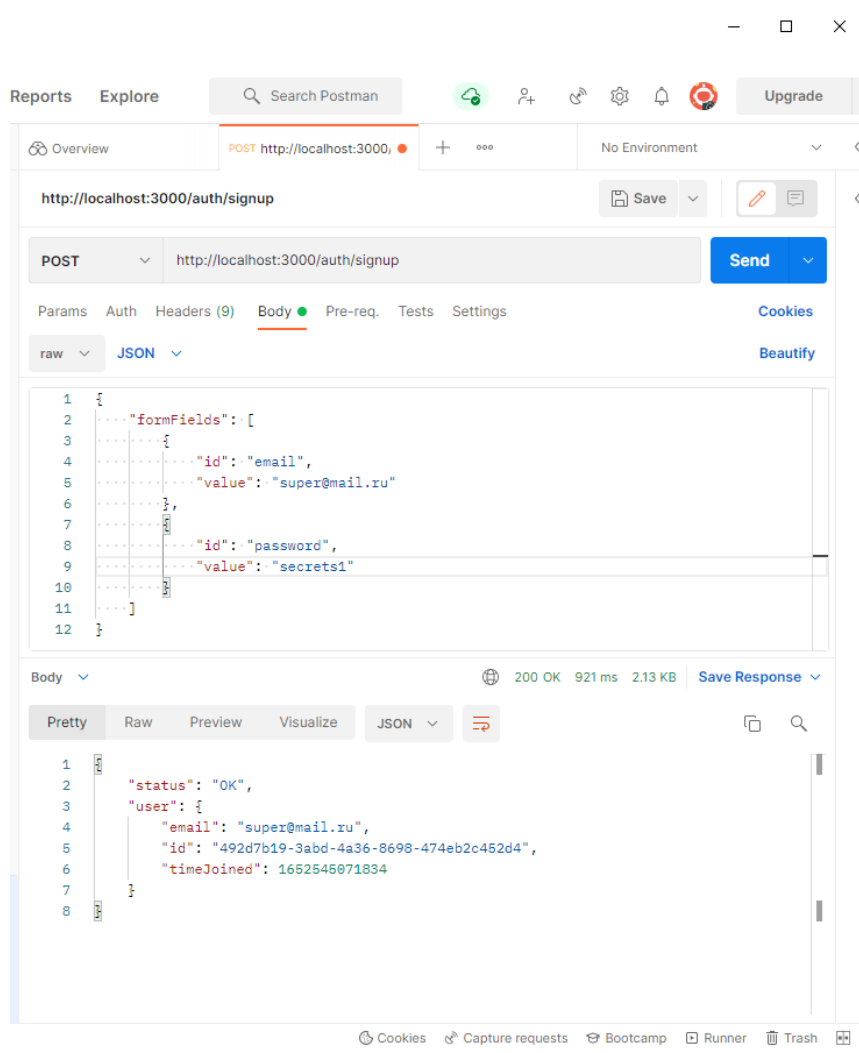
Для тестирования буду использовать postman.

Попробуем получить все девайсы:



Получили ответ “не авторизован”, тк метод защищен.

Теперь зарегистрируем пользователя и войдем.



GET http://localhost:3000/c

POST http://localhost:3000/

+

...

No Environment

http://localhost:3000/device

Save

GET

http://localhost:3000/device

Send

ParamsAuthorizationHeaders (7)BodyPre-request ScriptTestsSettingsCookies

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		

BodyCookies (2)Headers (9)Test Results

Status: 200 OKTime: 593 msSize: 2.19 KBSave Response

PrettyRawPreviewVisualizeJSON

18

"title": "Apple iPad 10,2 (2021) Wi-Fi + Cellular 256 Гб, серый космос",

19

"description": "Мощный. Простой в использовании. Универсальный и доступный. Создан специально для ваших любимых занятий. Работайте, играйте, творите, учитесь и делайте множество других дел.",

20

"price": 100500,

21

"basketId": null

22

},

23

{

24

"id": 6,

25

"title": "Apple iPhone 13 Pro Max, 256Гб, графитовый",

26

"description": "Просто. Нереально. Дисплей Super Retina XDR с технологией ProMotion и быстрым, плавным откликом. Грандиозный апгрейд системы камер, открывающий совершенно новые возможности. Исключительная прочность. A15 Bionic – самый быстрый чип для iPhone. И впечатляющее время работы без подзарядки. Всё это – в iPhone 13 Pro Max."

Все получилось, можем проверить куки.

GET http://localhost:3000/c

POST http://localhost:3000/

+

...

No Environment

http://localhost:3000/device

Save

GET

http://localhost:3000/device

Send

ParamsAuthorizationHeaders (7)BodyPre-request ScriptTestsSettingsCookies

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		

BodyCookies (2)Headers (9)Test Results

Status: 200 OKTime: 593 msSize: 2.19 KBSave Response

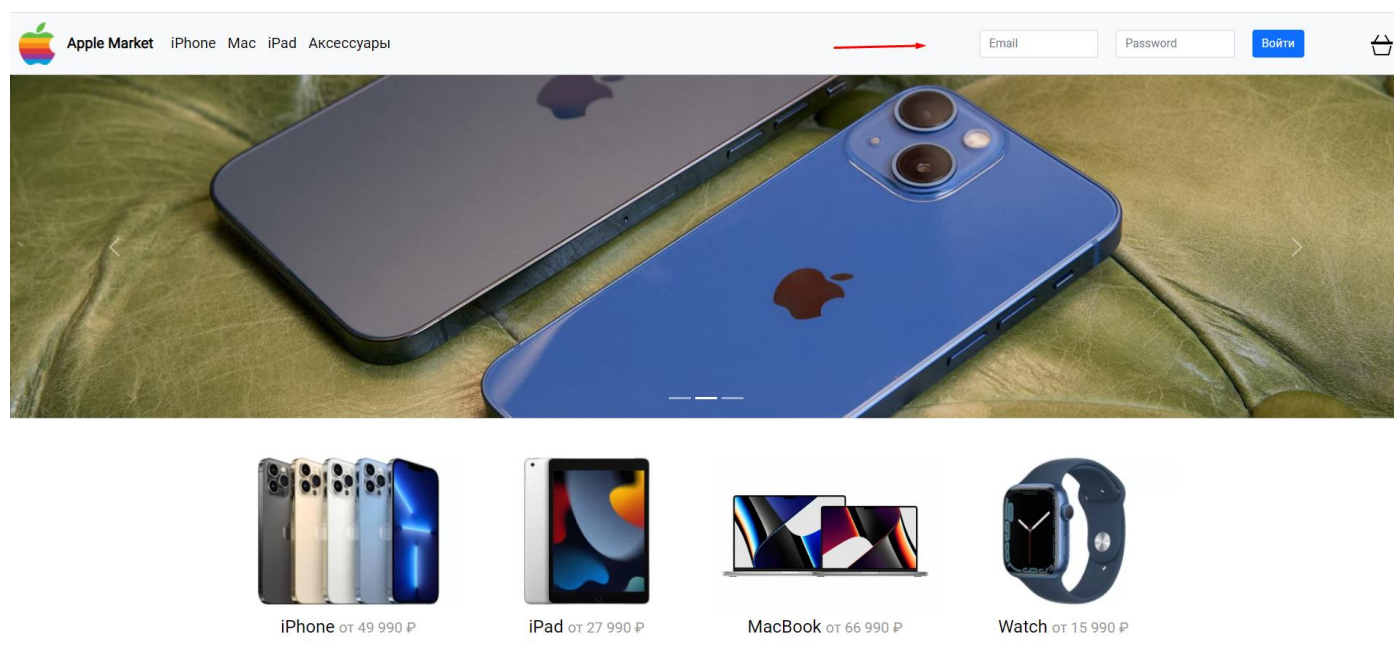
Name	Value	Domain	Path	Expires	HttpOnly	Secure
sAccessToken	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXLT...	localhost	/	Sun, 15 May '23 12:00:00 GMT	true	false
sIdRefreshToken	3876267d-bcf0-4b4a-8b4a-4b4a-4b4a-4b4a-4b4a-4b4a	localhost	/	Tue, 23 Aug '23 12:00:00 GMT	true	false

Теперь можем приступить к работе над фронтендом.

Создал файл auth.js для авторизации и выхода, подключил его ко всем страницам.

```
auth.filter.ts × supertokens.service.ts × auth.module.ts × auth.middleware.ts × main.ts × auth.guard.ts × session.decorator.ts ×
1 function login() {
2   const email = document.getElementById( elementId: 'email').value;
3   const pass = document.getElementById( elementId: 'pass').value;
4   console.log(email, pass);
5
6   fetch( input: 'http://localhost:3000/auth/signin', init: {
7     method: 'POST',
8     headers: {
9       Accept: 'application/json, text/plain, */*',
10      'Content-Type': 'application/json',
11    },
12    body: JSON.stringify( value: {
13      formFields: [
14        {
15          id: 'email',
16          value: email,
17        },
18        { id: 'password', value: pass },
19      ],
20    },
21  }) Promise<Response>
22    .then((response :T ) => console.log(response)) Promise<void>
23    .catch((err) => console.log(err));
24 }
```

Для входа надо использовать следующую форму.



После входа можем увидеть добавленный токен в куки.

В использовании.
доступный. Создан
ших любимых
е, играйте, творите,
ь и делайте множество

Выйти

Application

Manifest

Service Workers

Storage

Storage

Local Storage

Session Storage

IndexedDB

Web SQL

Cookies

http://localhost:3000

Trust Tokens

Interest Groups

Cache

Cache Storage

Back/forward cache

Background Services

Background Fetch

Background Sync

Notifications

Filter

Name	Value	T	F	I	S	I	S	S
sIRTFron...	5acf3c...	I..	/	2.	4..			L..
sFrontTo...	eyJ1a...	I..	/	9.	1..			L..
sIdRefre...	5acf3c...	I..	/	2.	5..	✓		S..
sAccessT...	eyJhb...	I..	/	2.	8..	✓		S..
Webstor...	65b5d...	I..	/	2.	5..	✓		S..