

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

Отчет

по лабораторной работе №2
по дисциплине «Web-программирование»

Автор: Шарапков Егор Викторович М33081

Преподаватель: Приискалов Роман Андреевич



УНИВЕРСИТЕТ ИТМО

Санкт-Петербург 2022

Для выполнения работы я выбрал шаблонизатор handlebars.

Шаблонизатор — программное обеспечение, позволяющее использовать шаблоны для генерации конечных документов с помощью декларативного языка разметки. Основная цель использования шаблонизаторов — это отделение формы документа и данных от полученного в результате документа. Использование шаблонизаторов улучшает читаемость кода и внесение изменений.

Я буду использовать шаблонизатор Handlebars.

Устанавливаем handlebars:

```
$ npm install --save hbs
```

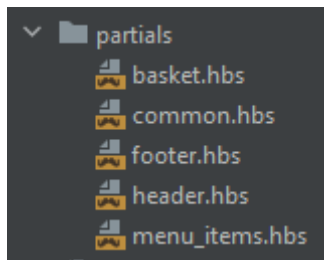


```
const app = await NestFactory.create<NestExpressApplication>(AppModule);

app.useStaticAssets(join(__dirname, '..', 'public'));
app.setBaseViewsDir(join(__dirname, '..', 'views'));
app.setViewEngine('hbs');
```

Мы сказали Express, что public каталог будет использоваться для хранения статических ресурсов, views будет содержать шаблоны, а hbs - механизм шаблонов, который должен использоваться для вывода HTML.

Далее создадим подкаталог views/partials, куда добавим частичные представления hbs.



Выделяем повторяющиеся части в отдельные представления, те части, которые используются на всех страницах, например header и footer.

Теперь мы можем вставить частичные представления в основные страницы.

Для вставки частичного представления применяется выражение `{{> name }}`, в котором прописывается имя файла частичного представления без расширения.

```
1 <!DOCTYPE html>
2 <html lang="ru">
3   <head>
4     {{> common }}
5     <link rel="stylesheet" href="/css/style.css">
6   </head>
7   <body>
8     <div class="wrapper">
9       {{> header }}
10      <div class="content">
```

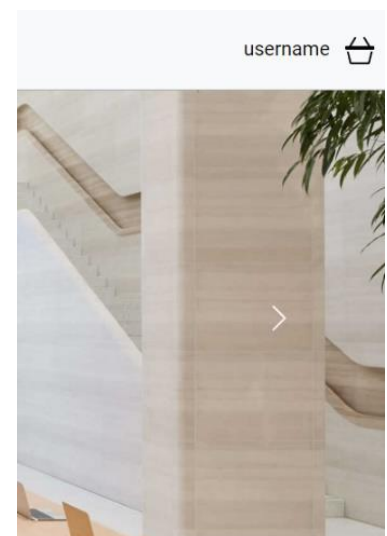
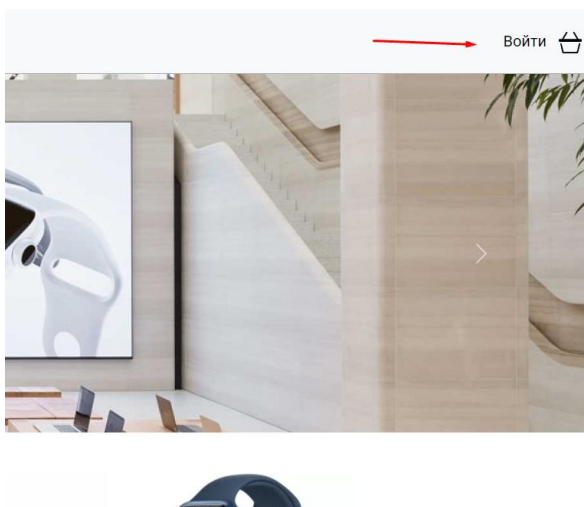
Я выделил следующие общие части: header (меню сайта), footer, common (основная метаинформация и подключение файлов скриптов и стилей), basket (модальное окно корзины), menu_items (перечень пунктов меню).

Для обеспечения рендеринга представлений добавляю маршруты для страниц.

```
main.ts x header.hbs x common.hbs x menu_items.hbs x app.controller.ts x footer.hbs x
1 import {Controller, Get, Render} from '@nestjs/common';
2
3 @Controller()
4 export class AppController {
5
6   @Get( path: ['/', '/index'])
7   @Render( template: 'index')
8   getIndexPage() {
9     return { isLoggedIn : Math.random() < 0.5 };
10  }
11
12   @Get( path: '/ipad')
13   @Render( template: 'ipad')
14   getIpadPage() {
15     return { isLoggedIn : Math.random() < 0.5 };
16  }
17
18   @Get( path: '/iphone')
19   @Render( template: 'iphone')
20   getIphonePage() {
21     return { isLoggedIn : Math.random() < 0.5 };
```

Для наглядности, с вероятностью 50% будет генерироваться страница либо с авторизованным пользователем, либо с не авторизованным.

Подготовил два состояния для представления информации о текущей сессии пользователя (авторизован и не авторизован).




Далее необходимо обновить поле, отвечающее за то, сколько времени потребовалось для отрисовки страницы, но, с включением того времени, которое понадобилось серверу для его обработки.

Для решения данной задачи имплементируем свой класс типа `Interceptor` и зарегистрируем его для выполнения перед каждым запросом внутри NestJS приложения.

```
@Injectable()
export class LoggingInterceptor implements NestInterceptor {
  intercept(context: ExecutionContext, next: CallHandler): Observable<any> {
    const now = Date.now();
    let ms = randomInt(100, 500);
    return next.handle().pipe(delay(ms)).pipe(
      tap( next: () => {
        hbs.registerHelper('time', function () {
          return Date.now() - now;
        })
      })
    );
  }
}
```

```
1 <footer class="footer text-center border-top p-4">
2   <div class="row">
3     <div class="col-12 pb-2">
4       <h5>Apple Market, Saint-Petersburg. <a href="support"> Поддержка</a></h5>
5     </div>
6     <h5 id="time">Total load time: <strong>{{ time }} ms</strong> (server) + </h5>
7   </div>
8 </footer>
```

Видим результат в футере.

Apple Market, Saint-Petersburg.  Поддержка
Total load time: **475 ms** (server) + **459 ms** (client)