# HTML шаблони

1. Подбираме в Интернет един или повече подходящи HTML шаблони.

В конкретния случай сме подбрали 3 шаблона. И трите са базирани на Bootstrap и са ориентирани към on-line търговия. Шаблоните са свалени в папка downloads.

1. Разархивирам шаблоните в папка „шаблони“.
2. В папката за шаблони създавам нов шаблон

Новият шаблон (папка „base“) ще бъде всъщност работният шаблон, върху който ще изграждаме нашето приложение. За основа вземам шаблона „famms“

* копирам всички файлове в новата папка.
* т.к. ще правим SDI приложение премахвам всички файлове от основната папка освен index.html

1. Променяме дизайна с цел да създадедем макет на нашето приложение
   1. Променяме менюто:
      * Махаме количката и бутона за търсене с цялата форма към него
      * Махаме точката „PAGES“ от менюто, т.к. тя е dropdown – с подменюта, и не ни трябва
      * Променяме наименованията на точките от менюто с оглед на функционалността, която целим.
   2. Променямe логото
   3. От папката „css“ махам файловете style.csss и style.css.map
   4. Прoменямe червенит цвят на зелен
2. Създавме секции за всеки функционален елемент на приложението:
   1. Начален екран – секции „Главно меню“ и „Слайдер“
   2. Списък обяви
   3. Влизане на регистриран потребител
   4. Регистриране на нов потребител
   5. Детайли за обява
   6. Списък подадени обяви и отзиви към тях на потребител
   7. Футер
3. Копираме готовия шаблон index.html в папка templates
4. Създаваме папка static и копираме в нея папките css, fonts, images и js.
5. Настройваме връзките към статичните ресурси:
   1. Добавяме директива „load static“:

<!DOCTYPE html>  
{% load static %}  
<html xmlns="http://www.w3.org/1999/html">

1. Добавяме директиви „static“ където е необходимо:

<link rel="stylesheet" type="text/css" href="{% static 'css/bootstrap.css'%}" />  
*<!-- font awesome style -->*<link href="{% static 'css/font-awesome.min.css' %}" rel="stylesheet" />  
*<!-- Custom styles for this template -->*<link href="{% static 'css/style.css' %}" rel="stylesheet" />  
*<!-- responsive style -->*<link href="{% static 'css/responsive.css' %}" rel="stylesheet" />

# VUE

1. Включваме необходимите „.js“ файлове за използване на VUE и axios:

<script src="https://unpkg.com/vue@3"></script>  
<script src="https://cdn.jsdelivr.net/npm/axios/dist/axios.min.js"></script>  
<script src="{% static 'js/logic.js' %}"></script>

1. Към тага body добавяме параметър id:

<body id="main">

1. Добавяме необходимите директиви за разделяне на действието на шаблонизатора Jinja между сървъра и клиента:

{% verbatim %}  
<body id="main">  
 ....  
 {% endverbatim %}  
 <a class="navbar-brand" href="index.html"><img width="250" src="{% static 'images/logo.png' %}" alt="#" /></a>  
 {% verbatim %}  
 ....

{% endverbatim %}  
 <img src="{% static 'images/slider-bg.jpg' %}" alt="">  
 {% verbatim %}

....

</div>  
{% endverbatim %}  
<!-- jQery -->  
<script src="js/jquery-3.4.1.min.js"></script>

1. Създаваме файла logic.js и генерираме минималното му съдържание – JavaScript обект по изискванията на VUE и обвързване на този обект с нашия шаблон:

const App = {  
 data() {  
 return {  
 status: 'начало', *// условно име на текущо визуализираната секция (или група секции)* }  
 },  
  
 methods: {  
 setSection(SectionName){  
 this.status=SectionName  
 },  
 },  
 created: function(){  
 this.status=0  
 }  
}  
  
Vue.createApp(App).mount('#main')