Професионална гимназия по електротехника и електроника —

гр. Пловдив

професия код 481030 „Приложен програмист“

специалност код 4810301 „Приложно програмиране“

ДОКЛАД

Тема: „Динамичен уеб сайт с база от данни“

Разработил: Петьо Петков №21, Сафет Терзиев №23, Петър Топалов №20; 11А клас

Ръководител-консултант: г-н Алекс Кьосев

ПЛОВДИВ, 2024 г.

СЪДЪРЖАНИЕ

ПРЕДМЕТ

Настоящият доклад, с тема „Динамичен уеб сайт с база от данни“, представлява назначена групова работа по учебната дисциплина „Разработка на софтуер“, с цел представянето и описанието на разработка с упоменатата тема, нейните особености, наложителни за разработка технологии, естетичен дизайн и систематичен набор от програмен код реализиращ я.

Горепосочените аспекти ще бъдат разяснени в същинската част на доклада, чрез подходящият словес, графики и снимки.

УВОД

Динамичните уеб сайтове са важна част от съвременната онлайн достъпност. Техните възможности за персонализация и интерактивност ги правят не просто носители на информация, а и активни платформи за масово взаимодействие. Чрез динамичния им характер те предоставят възможност за индивидуализирани потребителски преживявания, които удовлетворяват нуждите и предпочитанията на конкретния потребител.

Един от ключовите им аспекти е тяхната възможност за автоматизирано обновяване на съдържанието. Това означава, че информацията може да бъде актуализирана, базирана на различни фактори като потребителските предпочитания, действията на потребителите или външни събития. Тази автоматизация не само прави уеб сайтовете по-актуални и реактивни към промените в околната среда, но също така осигурява по-ефективно управление на съдържанието и по-голяма ефективност в работата.

Освен това, динамичните уеб сайтове предоставят богат набор от мултимедийни и интерактивни възможности. Те могат да интегрират различни видове съдържание като видео, аудио, и други елементи, което не само ги прави по-привлекателни за потребителите, но и предоставя по-богато и ангажиращо потребителско преживяване.

В крайна сметка, динамичните уеб сайтове не само предоставят по-богато и атрактивно потребителско преживяване, но и играят ключова роля в успеха на онлайн бизнесите, като подобряват ангажираността на потребителите и предоставят ценни данни за бизнес анализ и оптимизация.

ИЗПОЛЗВАНИ ТЕХНОЛОГИИ

За реализацията на проекта са използвани технологиите HTML, CSS, JavaScript, Firebase, git и GitHub, които биват комбинирани и употребени във Visual Studio Code.

Всяка от следните технологии играе ключова роля за споменатата задача, като HTML и CSS осигуряват структурата, стила и възможността за наличие и добавяне на същинско съдържание, а JavaScript осигурява динамичната и интерактивната характерност на уеб сайта както и възможността за работа с функциониращата база от данни, предоставена и създадена чрез технологията Firebase. Платформата GitHub предоставя възможността както за улесняването за интеграция на доброкачествени промени по разработката, така и ускоряването на работният ѝ процес. Системата, осигуряваща гореспоменатите черти на платформата е git.

HTML е основният език за създаване и структуриране на уеб сайтове. Той използва различни елементи и тагове (markup tags), които определят съдържанието и структурата на уеб сайта. Те се използват за маркиране на различни видове съдържание като текст, изображения, връзки, форми и други елементи.

CSS е език за стилизиране на уеб сайтовете. Той се използва за дефиниране на външния вид и оформлението на елементите, включително текст, изображения, различни контейнери и други. Чрез CSS, уеб разработчиците могат да контролират различни аспекти на визуалното представяне на уеб сайтовете, като цветове, шрифтове, размери, отстъпи и много други.

JavaScript е високо ниво интерпретиран програмен език. Той се използва главно за уеб разработка, за добавяне на интерактивност към уеб сайтовете. JavaScript позволява на уеб разработчиците да създават динамични ефекти, интерактивни функции и сложна логика, която да се изпълнява директно в уеб браузъра на потребителя.

Firebase е платформа за разработка на мобилни и уеб приложения, предоставяща разнообразни инструменти и услуги, които улесняват и ускоряват процеса. Тази платформа предлага набор от инструменти за създаване на уеб приложения, анализиране на данни, управление на потребители и много други.

Git е децентрализирана система за контрол на версиите на файлове.

GitHub е уеб базирана услуга за разполагане на софтуерни проекти и техни съвместни разработки върху отдалечен интернет сървър в т.нар. хранилище. Базира се на Git системите за контрол и управление на версиите.

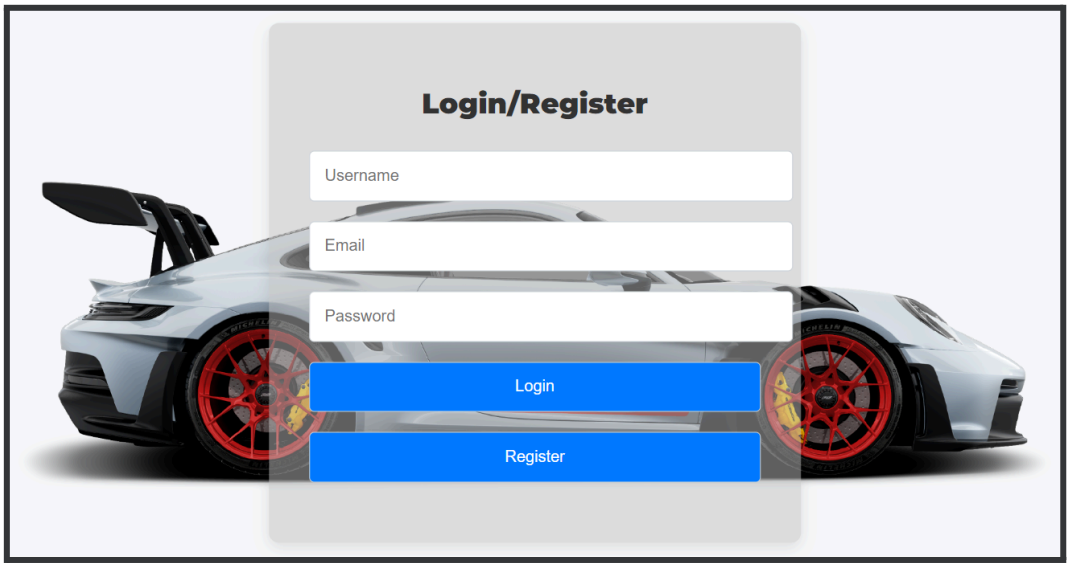
РАЗРАБОТКА

Разработката представлява примерна Login/Register система, подходяща и належаща за всеки един динамичен уеб сайт, ползващ се с регистрирани в него потребители, представена за клиента чрез login/register форма, имаща опции за създаване (регистриране) на потребителски профил чрез подходящи данни, които биват валидирани от своя страна за постигането на безопасна кибер среда както и влизане в системата спрямо валидни входни данни, вече съществуващи като релационни записи в присъщата на разработката база от данни, и предоставяне на достъп към цялостното клиентско преживяване.

От тук следва и постигането на цялостна идентификация и автентикация на потребителя.

След успешно влизане на потребителя, му се визуализира началната страница със съдържание, откъдето сайта може да бъде надграждан чрез нови страници и съответно ново съдържание и услуги или промяна на вече съществуващите такива.

За удобство на четеца и последващата частна описателна характерност на словеса ще обозначим страницата за създаване и вход на потребител (Login/Register формата) като „страница №1“, а началната страница като „страница №2“.



Фиг. 1. Страница №1.

На Фиг. 1 се наблюдава какво вижда клиента, след като влезе в уеб сайта. Това е именно Login/Register формата, която той трябва да попълни, за да може да се

регистрира или да влезе в своя вече съществуващ акаунт. Тя се състои от три различни текстови полета, всяко от които отговаря съответно за въвеждането на потребителско име, имейл и парола както и от два бутона Login за влизане и Register за регистрация на нов потребител. Необходимо е да въведат данни и за трите полета, след което да се натисне подходящия бутон.

Фиг. 2. Страница №2

На фиг. 2 се наблюдава резултатът от успешното влизане на потребителя, а именно визуализацията на началната страница. Тя представлява статична страница с графични и текстови елементи. А от тук вече започва и същинската клиентска интерактивност с уеб сайта и на потребителя му се предоставя възможността да достъпи до всяка една от предназначените за него услуги.

ПРАКТИЧЕСКА ОБОСНОВКА

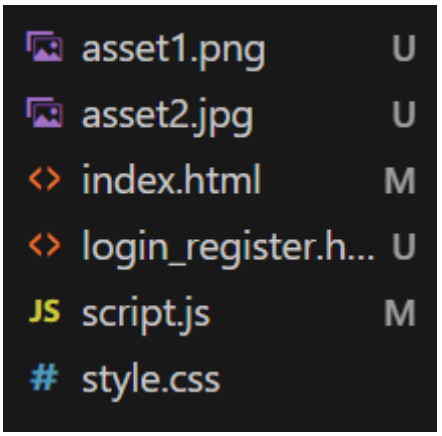
Тук вече ще представим самата разработка от към чисто техническата и реализационната ѝ страна. Ще обърнем внимание както на файловата, така и на кодовата структура на проекта, а самият код нужен за реализацията ще бъде разгледан функционално чрез пояснения.

1. Структрура.

1.1. Файлова структура.

Проектът се състои от 6 файла, два от тях биват графичен ресурс, които са неизменна част от съдържанието на уеб сайта. Останалите четири са същинските програмни файлове, като два от тях са .html файлове, отговорни за конструктурния вид на сайта, именно страница №1(login_register.html) и страница №2(index.html).

Налични са един .css файл, отговорен за цялостната визуална стилизация на уеб сайта, спрямо неговите характерни елементи и желан от разработчиците дизайн както и един .js файл, осигуряващ сайтовата фунцкионалност, работата и връзката с базата от данни.



Фиг. 3. Файлова структура на проекта.

1.2. Кодова структура.

Ролево кодът се разделя на две части, като едната отговаря за конструктурния вид на сайта и неговото съдържание, а другата за функционалните му услуги и динамичност.

2. Практическа реализация и пояснения

2.1. Конструкционна част на сайта и съдържание.

Двата файла, отговорни за тази роля биват login_register.html(страница №1) index.html(страница №2).

2.1.1. Login/Register форма.

```
3     <head>
4         <meta charset="utf-8">
5
6         <title>WebsiteSD</title>
7
8         <link rel="preconnect" href="https://fonts.gstatic.com">
9         <link href="https://fonts.googleapis.com/css2?family=Montserrat&display=swap" rel="stylesheet">
10        <link href="https://fonts.googleapis.com/css2?family=Montserrat:wght@900&display=swap" rel="stylesheet">
11        <link href="https://fonts.googleapis.com/css2?family=Montserrat:wght@500&display=swap" rel="stylesheet">
12        <link href="https://fonts.googleapis.com/css2?family=Bowlby+One+SC&display=swap" rel="stylesheet">
13
14        <link rel="stylesheet" type="text/css" href="style.css">
15    </head>
```

Head елементът се състои от множество от link елементи, отговорни за наличието на готови шрифтове(редове 7 до 8) и един предоставящ връзката на страницата с .css файла, оформящ визуалната стилизация(ред 14).

На ред 6 се задава заглавието на страницата чрез title елемента.

```
17      <body>
18          <div id="content_container">
19              <div id="form_container">
20                  <div id="form_header_container">
21                      <h2 id="form_header">Login/Register</h2>
22                  </div>
23
24                  <div id="form_content_container">
```

В главната част на страницата или body елемента са налични множество от div(от англ. „сектор“) елементи, отговорни за структурирането на съдържанието.

Главният сектор с id “content_container” е този(ред 18), който помества в себе си цялото съдържание на страницата.

Следва “form_container” сектора(ред 19), в който се помества самата login/register форма. А сектора на ред 20 осигурява наличието на заглавие на формата, в случая „Login/Register“. От ред 24 започва сектора за съдържанието на формата.

```
24                  <div id="form_content_container">
25
26                      <div id="form_content_inner_container">
27                          <input type="text" id="full_name" placeholder="Username">
28                          <input type="email" id="email" placeholder="Email">
29                          <input type="password" id="password" placeholder="Password">
30
31                          <div id="button_container">
32                              <button onclick="login()">Login</button>
33                              <button onclick="register()">Register</button>
34                          </div>
35
36                      </div>
37                  </div>
```

Самата форма за въвеждане на данни се реализира от трите input елемента на редове 27 до 29, като всеки от тях отговаря съответно за потребителско име, имейл и парола.

На ред 31 се реализира сектора, в който се поместват двата бутона за влизане с име „Login“ и регистриране с име „Register“, снабдени с динамична характерност, при тяхната употреба от клиента.

```
40      </body>
41
42      <script src="https://www.gstatic.com/firebasejs/8.6.8/firebase-app.js"></script>
43      <script src="https://www.gstatic.com/firebasejs/8.6.8/firebase-auth.js"></script>
44      <script src="https://www.gstatic.com/firebasejs/8.6.8/firebase-database.js"></script>
45
46      <script src="script.js"></script>
47
48  </html>
```

След завършването на body елемента се реализира връзката с главния файл за снабдяване на функционалност на сайта(ред 46) — script.js. И тази с базата от данни, създадена и предоставена от платформата Firebase(редове 42 до 44).

2.1.2. Основно съдържание.

```
1   <!DOCTYPE html>
2   <html lang="en">
3       <head>
4           <meta charset="UTF-8">
5           <meta name="viewport" content="width=device-width, initial-scale=1.0">
6           <title>Welcome to Your App</title>
7
8           <link href="style.css" rel="stylesheet">
9       </head>
10
11      <body id="index-body">
12          <h1 id="index-h1-undercover">ОГЛЕДАЙ СЕ, ВИЖДАШ ЛИ НЯКЪДЕ МОРАЛ ТУКА, БЕ?</h1>
13          <h3 id="index-h3-login-confirmation">Ти се логна</h3>
14          <img src="asset1.jpg" alt="Opel" id="index-opel_image">
15      </body>
16  </html>
```

Файлът index.html се състои от един head и едно body. В head-а се съдържа метаинформацията за документа като настройките за символен набор и заглавие. В body се определят стиловете за елемента body, като шрифтове и текстово оформление. h1 задава стилове за заглавните елементи от първо ниво, а img вмъква необходимият графичен ресурс.

2.2. Функционална част на сайта.

```
1
2   const firebaseConfig = {
3     apiKey: "AIzaSyCcSwxd4reb8Xj4qX7NRA4ts9FZb9SkpzQ",
4     authDomain: "loginregister-d7a0a.firebaseapp.com",
5     projectId: "loginregister-d7a0a",
6     storageBucket: "loginregister-d7a0a.appspot.com",
7     messagingSenderId: "569803298341",
8     appId: "1:569803298341:web:3a78f44136969e958e91cc",
9     measurementId: "G-ZCZF0VFKS7"
10  };
11
12  firebase.initializeApp(firebaseConfig);
```

Настройка за конфигурацията на Firebase (редове 1 до 12).

Тук се инициализира Firebase, използвайки предоставения обект за конфигурация firebaseConfig, който съдържа API ключове и друга необходима информация за връзка с услугата Firebase.

```
15  function register () {
16    email = document.getElementById('email').value
17    password = document.getElementById('password').value
18    full_name = document.getElementById('full_name').value
19
20    if (validate_email(email) == false || validate_password(password) == false) {
21      alert('Email or Password is Outta Line!!')
22      return
23    }
24    if (validate_field(full_name) == false) {
25      alert('One or More Extra Fields is Outta Line!!')
26      return
27    }
28
29    auth.createUserWithEmailAndPassword(email, password)
30    .then(function() {
31      var user = auth.currentUser
32
33      var database_ref = database.ref()
34
35      var user_data = {
36        email : email,
37        full_name : full_name,
38        last_login : Date.now()
39      }
40
41      database_ref.child('users/' + user.uid).set(user_data)
```

```
42
43        alert('User Created!!')
44      })
45      .catch(function(error) {
46        var error_code = error.code
47        var error_message = error.message
48
49        alert(error_message)
50      })
51    }
```

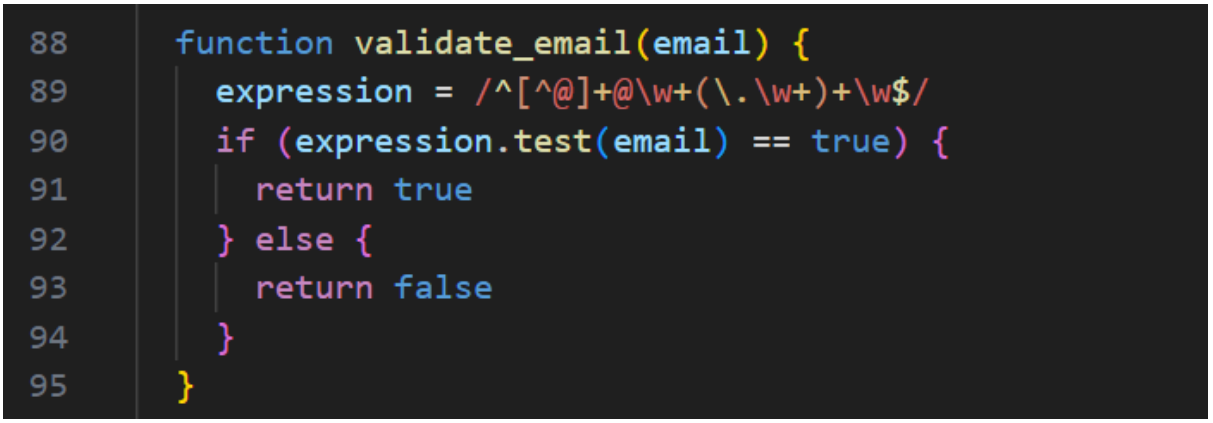
Функция за регистрация (редове 14 до 51).

Този блок дефинира функцията register(). Тя събира данните на потребителя (име, имейл, парола) от HTML формат, валидира полетата, след което създава нов потребител, използвайки метода createUserWithEmailAndPassword на Firebase. При успешна регистрация, тя съхранява данните на потребителя за последното влизане в Firebase Realtime Database под възела users с уникалния идентификатор на потребителя. Ако регистрацията се провали, извежда съобщение за грешка на потребителя.

```
53  function login () {
54    email = document.getElementById('email').value
55    password = document.getElementById('password').value
56
57    if (validate_email(email) == false || validate_password(password) == false) {
58      alert('Email or Password is Outta Line!!')
59      return
60    }
61
62
63    auth.signInWithEmailAndPassword(email, password)
64    .then(function() {
65      var user = auth.currentUser
66
67      var database_ref = database.ref()
68
69      var user_data = {
70        last_login : Date.now()
71      }
72
73      database_ref.child('users/' + user.uid).update(user_data)
74
75      alert('User Logged In!!')
76
77        window.location.href = "index.html";
78    })
79    .catch(function(error) {
80      var error_code = error.code
81      var error_message = error.message
```

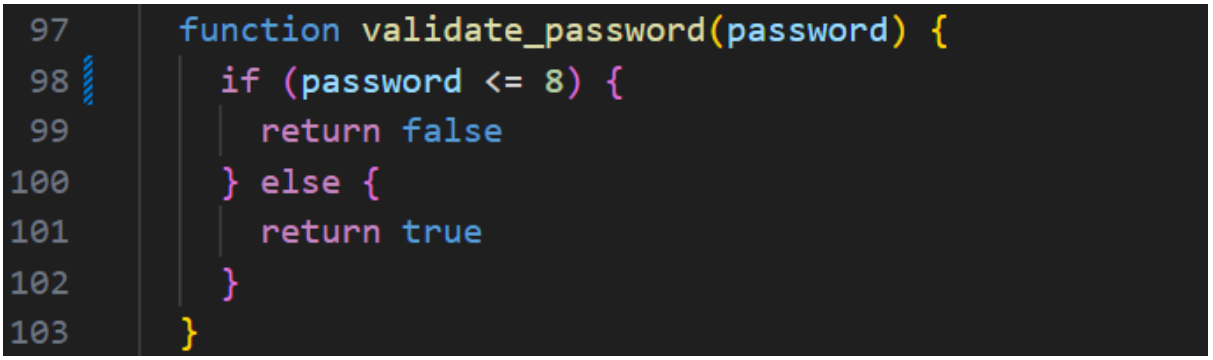
Функция за вход (редове 53 до 81).

Тук се помества функцията login(). Подобно на функцията за регистрация, тя събира входа на потребителя (имейл и парола) от HTML формата, валидира полетата за имейл и парола, след което се опитва да входира потребителя, използвайки метода signInWithEmailAndPassword() предоставен от Firebase. При успешен вход, актуализира времевия сегмент относно последното влизане на потребителя в базата от данни под върха users с уникалния идентификатор на потребителя. Допълнително пренасочва потребителя към страницата index.html. Ако входът се провали, извежда съобщение за грешка на потребителя.



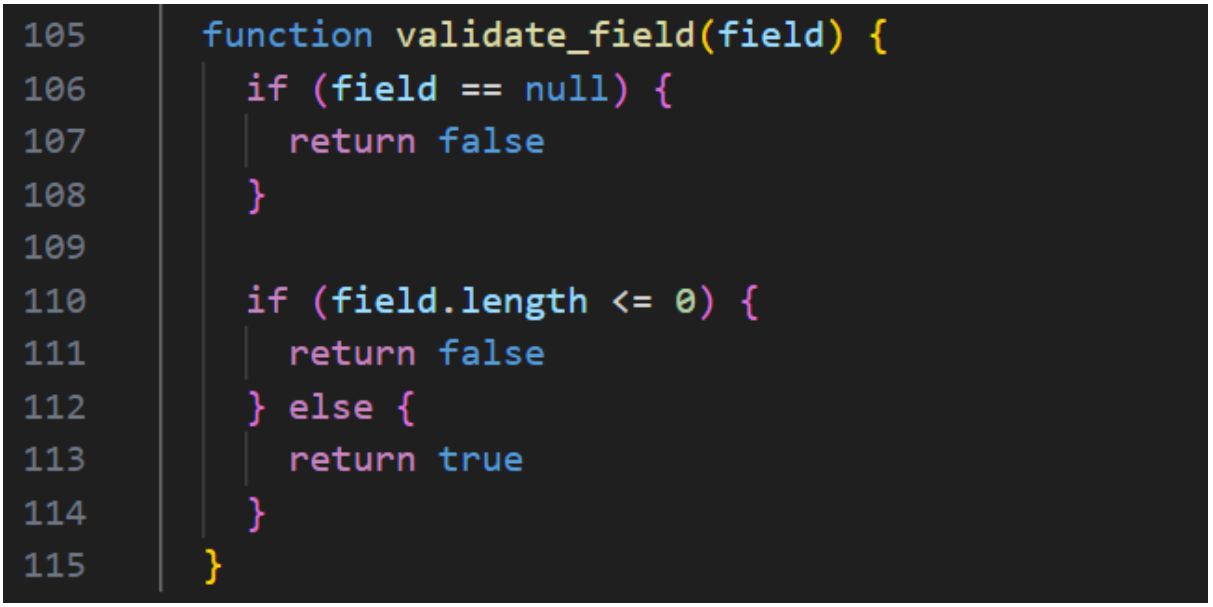
Функция за валидация на имейл(редове 88 до 95)

Тук се дефинира функцията validate_email(). Приема низ за имейл като вход и проверява дали той съвпада с регулярен израз за валиден формат на имейл. Ако имейлът отговаря на шаблона, то тогава имейлът е валиден, в противен случай имейлът не се приема.



Функция за валидация на парола(редове 97 до 103)

В този блок се дефинира функцията validate_password(). Приема низ за парола като вход и проверява дали дължината е по - голяма от 8 символа. Ако паролата отговаря на изискванията за дължина, тя се приема, в противен случай паролата става невалидна.



```
105  function validate_field(field) {
106    if (field == null) {
107      return false
108    }
109
110    if (field.length <= 0) {
111      return false
112    } else {
113      return true
114    }
115  }
```

Функция за валидация на поле(редове 105 до 115)

Тук се дефинира функцията validate_field(). Приема поле (например, име) като вход и проверява дали то е не е празно и има дължина по–голяма от 0 символа. Ако полето в валидно то се приема, в противен случай то не се приема.

ЗАКЛЮЧЕНИЕ

Динамичните уеб сайтове представляват важен инструмент за създаване на богато, интерактивно и персонализирано онлайн преживяване за потребителите, като е необходима балансирана подходяща архитектура и поддръжка за тяхната успешна реализация.

Представената в настоящия доклад разработка отговаря на новопроектиран динамичен уеб сайт, имащ функционираща клиентска и сървърна част и от там съответно възможността за надграждане и цялостна реализация на един завършен и готов за масова употреба продукт.

За тези, които се интересуват от практическата реализация на разработката, е предоставен цялостния проект в приложение №1.

ПРИЛОЖЕНИЯ

1. Цялостна разработка в GitHub
   <https://github.com/Artreus/Website_SD-21-20-23.git>