5BII 2019-2020 Davide Cani

Creazione di un sito web da zero: Progetto Berghem Bike

INDICE

1. Introduzione

1.1. Progetto Berghem Bike

Berghem Bike è un progetto che ho ideato per apprendere le basi per lo sviluppo di siti web in stile tradizionale e anche di alcune tecniche moderne.

All'inizio dello sviluppo ho stabilito questi obiettivi:

- Lavorare con molteplici linguaggi di programmazione facendoli lavorare in cooperazione.
- Creazione di un framework Frontend(TS) e Backend (PHP) riutilizzabile per futuri progetti
- Mettere in pratica le mie conoscienze teoriche di SQL
- Migliorare le mie competenze in ambito grafico con CSS

2. Design

2.1. Business plan

Executive summary

Concetto di business

Berghem Bike è un servizio di **bike sharing** che mette a disposizione biciclette per l'uso condiviso e propone un alternativa a basso costo e sostenibile per viaggi a corto raggio diminuendo il carico dei mezzi di trasporto pubblici , auto e moto.
Berghem Bike è un servizio per il cittadino, lo studente e il turista che vogliono spostarsi da punto A a punto B in modo pratico ed efficente.

Caratteristiche finanziarie

Per sostenere questa iniziativa con prezzi competitivi e abbassare il costo del servizio anche agli utenti. Verranno utilizzati degli sponsor pubblicitari mediante le biciclette e le stazioni di deposito. L'utile del servizio potrebbe essere reinvestito per espandere la rete delle stazioni e raggiungere anche i paesi limitrofi, migliorando e aumentando i potenziali acquirenti che usufruiranno del servizio.

Requisiti finanziari

Il capitale richiesto sarà concordato con il comune di Bergamo in collaborazione con le diverse aziende che forniranno i materiali necessari.
Il capitale verrà utilizzato per le seguenti componenti:

- Biciclette (50) per ciascuna stazione
- 50 slot per ciascuna stazione, (Arduino, lettore RFID, blocco meccanico elettronicamente comandato).
- Un totem per stazione per il ritiro della bicicletta (computer, collegamento internet con rete mobile, schermo, webcam per leggere codice QR da app del cellulare, monitoraggio e controller slot).
- Un totem informativo e pubblicitario per stazione
- Sito web
- Applicazione per cellulare che genera codice QR per accedere alle bici
- Applicativo per gestione abbonamenti e manutenzione del sistema.

Principali risultati:

Rendere il cittadino più consapevole della propria impronta sull'ambiente e incentivarlo a utilizzare un servizio sostenibile.

Piano di marketing:

La sostenibilità, il prezzo e la praticità del servizio sono i punti chiave che rendono il bike sharing avvincente.

Berghem Bike vuole affermarsi come il mezzo di trasporto più naturale da usare sia per il cittadino che per il turista occasionale.
Per assicurarsi la fiducia del cliente e incentivare l'utilizzo del servizio, si possono offrire dei premi come buoni cultura o buoni sconto in base a delle mete raggiunte (come il numero di ore di utilizzo o tramite la stima delle emissioni evitate)

Con praticità del servizio, si intende che si può effettuare la registrazione completamente online fornendo i propri dati anagrafici e confermando la propria identità con una foto della propria carta di identità.

Una volta caricato il credito sul proprio account si può usufruire del servizio da subito tramite un applicazione che genera un codice QR univoco per ogni viaggio che verrà letto dal totem.

Il prezzo del servizio, più basso rispetto ad altri bike sharing, è tale perchè i costi del servizio verranno sostenuti in parte grazie agli sponsor.
Il servizio verrà promosso dalla provincia, e per auto sostenersi le biciclette e le stazioni sponsorizzeranno aziende e altre attività locali.
La campagna pubblicitaria verrà effettuata tramite i social network più utilizzati, creando contenuti come video, foto, e iniziative che coinvolgano gli utenti.

2.2. Sviluppo web tradizionale e moderno

Fino al decennio scorso, le prestazioni dei dispositivi client-side erano limitate (Cellulari, tablet, computer datati) per rimediare si utilizzavano server ad alte prestazioni, che gestivano la logica e la navigazione del sito attraverso singole pagine a costo di un maggior numero di richieste.
Con lo sviluppo tradizionale (**multi page application**) il Frontend è quasi assente, era utilizzato solo per controllare l'immissione dati delle form prima di inoltrare al server.

Con l'avanzamento della tecnologia questa soluzione è diventata obsoleta, ora ci troviamo nella situazione opposta, dove quasi tutti i client hanno altissime prestazioni. È il motivo per cui lo sviluppo web è cambiato radicalmente in questi ultimi anni.

I Framework **Frontend** (come ad esempio Angular JS o Vue JS) approfittano di questo aspetto. Questi hanno implementato un modello di sviluppo modulare e facile da manterenere utilizzando un formato ibrido che contiene codice HTML e JavaScript per sviluppare le singole pagine. Le pagine vengono poi compilate e impacchette in una **single page application**. Una single page application gestisce la logica e la navigazione di più pagine interamente tramite il Frontend senza eseguire ulteriori richieste al server.

Lo sviluppo moderno è vantaggioso per il server: Permette di gestire un utenza molto più grande utilizzando molte meno risorse, sfruttando la potenza computazionale dei client.

La tecniche di sviluppo utilizzate in questo progetto sono sia tradizionali che moderne
Il Backend è gestito in maniera tradizionale con il linguaggio PHP, mentre il Frontend è gestito con alcune pratiche moderne ed è scritto in linguaggio **Typescript.**

Typescript è un lingaggio inventato da **Microsoft**, viene utilizzato in uno dei Framework JavaScript più utilizzati "**Angular JS**".

In questi ultimi anni si sta utilizzando sempre di più perchè offre un enorme vantaggio, uno strato di compilazione che introduce il "**Type Checking**" a JavaScript. Come il termine suggerisce, il Type Checking essenzialmente controlla che le variabili utilizzate dal programma non siano **undefined** o **null** e che si accedano in modo sicuro.
La compilazione viene annullata nel caso ci fosse un errore di compilazione, esattamente come nel linguaggio C/C++.

il codice scritto da me in TypeScript segue lo standard **ECMAScript 6** uscito nel 2015 che definisce la nuova sintassi standard.
Nomino alcuni degli aspetti più importanti:

- Definisce la nuova sintassi per le classi
- Stabilisce come esportare ed importare moduli e classi.
- Aggiunge le "Arrow function", una scorciatoia per creare una funzioni anonime che include il parametro contestuale "this"
- Aggiunge dei nuovi tipi di loop, variabili locali, e costanti.

Pillola di storia:
Per moltissimi anni JavaScript non aveva mai avuto uno standard, ogni progetto utilizzava diversi metodi per raggiungere gli stessi obiettivi.

Infine, un altra tecnica che ho implementato durante lo sviluppo è il **Bundling** che consiste nell'impacchettare il codice JavaScript compilato in un singolo file.
Un "**Bundler**" è uno strumento che riunisce il codice e tutte le sue dipendenze in un unico file JavaScript, riducendo ad uno il numero di file JS richiesti dal client.
Questa tecnica mi ha permesso di utilizzare la libreria di JavaScript per le mappe "Leaflet", senza Bundler non sarebbe compatibile con il mio codice essendo che utilizza uno standard diverso dal mio per esportare le classi (CommonJS).

2.3. Il Back-end e il Front-end

Il termine **Back-End** fa riferimento al "**data access layer**"
(livello di accesso dei dati) di un software, nel caso di un sito web fa riferimento al server, che gestisce le richieste ricevute dai client.

il **Front-End** fa riferimento al "**Presentation Layer**" cioè il livello di presentazione che viene visualizzato all'utente, in altri termini la pagina web che visualizza con il browser.

Ho scelto di sviluppare il Backend in linguaggio **PHP**, costruendo un sistema di gestione utenti orientato ad oggetti.
Tutte le richieste dei client vengono gestite attraverso un unico file chiamato "**action.php**" Quest'ultimo contiene anche delle sicurezze contro alcuni tipi di attacchi informatici che prendono di mira le debolezze del sito.

In quanto al Frontend, ho scelto il linguaggio **Typescript**, con il quale ho costruito un Framework che mi permette di gestisce la grafica mobile, le notifiche, il controllo delle form, la gestione dell'utente e delle richieste AJAX.

Nel prossimo capitolo verrà spiegato tramite il diagramma classi come Frontend e Backend interagiscono fra loro.

2.4. UML Diagramma Classi

I Diagrammi delle classi sono schemi di progettazione che aiutano gli sviluppatori a visualizzare e realizzare un sistema orientato ad oggetti.
Realizzare un diagramma classi prima dell'effettivo sviluppo di un Framework offre enormi vantaggi nel lungo termine:

- Permette di analizzare i requisiti e trovare la soluzione più efficiente
- Aiuta nel pianificare e dividere meglio i compiti all'interno di un team
- Ottimizza e diminuisce la quantità di lavoro, dividendo il design dalla realizzazione effettiva.

Il diagramma classi mi ha aiutato a realizzare i Framework e di modificarli durante lo sviluppo per adattarli meglio alle funzionalità richieste.

Qui riporto una versione semplificata del diagramma

Versione completa:

Tramite il diagramma si può dedurre in che modo avviene la comunicazione client-server, qui sono riportati gli esempi più importanti.

RICHIESTE CLIENT	RISPOSTE SERVER	REAZIONE DEL CLIENT DOPO RISPOSTA
Richiede la connessione	Trasmette file HTML, App.JS (Bundle) e CSS	Carica il Frontend, inizializza UIManager
UserManager richiede attraverso AjaxManager se l'utente è autenticato	Risponde con delle costanti, ALREADY_LOGGED_IN oppure NOT_LOGGED_IN.	Cambia il menù in base al risultato e salva booleano in sessione browser
Tenta di accedere trasmettendo le credenziali (sempre tramite user e ajax)	Risponde con delle costanti, SUCCESS, oppure codice di errore, non suggerisce se l'utente esiste o meno.	Mostra una notifica, mobile/desktop con il risultato.
Entra nella bacheca, UserManager carica DashManager che procede con la richiesta della pagina riepilogo	Dopo aver effettuato alcuni controlli, trasmette la pagina in HTML	Visualizza la pagina trasmessa visualizzandola all'interno del contenitore
Sempre nella bacheca, l'utente richiede di visualizzare le attività, prima manda una richiesta per la pagina, poi una richiesta per i dati.	**1**° richiesta manda la pagina attività in HTML **2**° richiesta manda i dati da visualizzare in XML	**1**° risposta Visualizza la pagina e attiva la gif di caricamento **2**° risposta sostituisce la gif di caricamento con la tabella che costruisce tramite l'XML

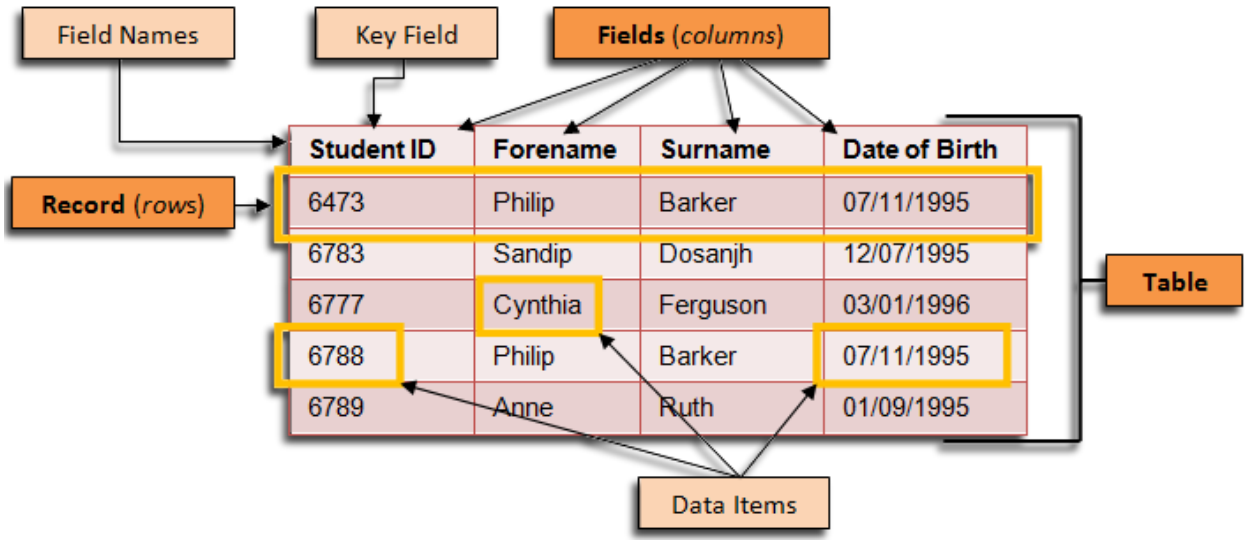
3. Sistemi e Reti, Informatica e TPS

3.1. (INGLESE) What is a Database?

Databases are digital filing systems which are used to store records, they are used for a variety of different applications in the digital industry.

Databases can hold very large amounts of data, for example user information, mailing lists or in the case of Google, huge lists of websites indexed on their search engine.
Information stored in the database can be retrieved through a "query".
Queries are a way to ask the database to find records which meet the conditions defined within the query.

Now, speaking on the technical side.



Records are stored inside tables as rows, tables have a very specific structure which is defined by the developer based on the requirements of the project, databases are built and organized upon them.
Records can contain multiple columns called fields of different data types.

There's one thing every record has in common, they all have a "Primary Key".
A primary key has to be unique for each record, it can be a number such as an ID or it can also be a string (a sequence of characters).
When defining a table, the developer also defines the structure of the record it's going to hold.
One very important aspect of tables is that a record can contain "Foreign keys" which are references that point to the Primary keys of records stored in different tables.
Foreign keys are a necessity in complex databases because they avoid the duplication of data.
As my teacher said a hundred times, a database is considered well made when it is not redundant.

3.2. Gestione del database con PDO

PHP 7 fornisce al programmatore una classe chiamata PDO (PHP Data Objects).

PDO è un "Database Access Abstraction Layer" cioè un livello di astrazione per la connessione e le chiamate al database, permette di evitare operazioni di base che altrimenti devono essere ripetute centinaia di volte in ogni applicazione.

Quando PDO è utilizzato correttamente permette di aumentare la sicurezza del Backend prendendosi cura della sanitizzazione delle stringhe prima di una query e fornendo una connessione unica al database.

Utilizzare la classe PDO offre enormi vantaggi:

- **Sicurezza**
  Tramite i "Prepared statements" e la sanitizzazione delle stringhe.
- **Facilità d'uso**
  Molte funzioni di supporto per automatizzare le operazioni di routine
- **Riusabilità**
  API unificata per accedere a moltissimi database, da SQLite a Oracle

Solitamente il PDO è gestito da un “Wrapper” per una questione di praticità, un wrapper è una classe che gestisce l’istanza di un altra classe.
in questo progetto il wrapper di PDO è chiamato **DbManager**
La classe DbManager gestisce l’accesso al database e le query associate

I Prepared Statements sono il punto forte della classe PDO
Un prepared statement è un metodo per inoltrare la richiesta al database, la query e i dati vengono inviati al server separati l'uno dall'altro e quindi non vi è alcuna possibilità che interferiscano. Il che rende assolutamente impossibile l’SQL Injection.

Tramite la funzione **prepare** si prepara la query da eseguire inserendo il carattere “ **?** ” al posto di dove vorremmo le nostre variabili.
Questo carattere si chiama **positional placeholder.**
ogni placeholder viene sostituito con il valore corrispondente all’interno dell’array che viene passato alla funzione **execute.**

Prendo come esempio la funzione query_login in DbManager per dimostrare il funzionamento di PDO durante l’autenticazione dell’utente

```php
public static function query_login($user)
{
    $statement  = null;
    $result     = null;
    if($user->getUsername() != null)   //if username
    {
        $statement = self::$pdo->prepare("SELECT * FROM users WHERE username = ?");
        $result = $statement->execute([$user->getUsername()]);
    }
    else if ($user->getEmail() != null) // if email
    {
         $statement = self::$pdo->prepare("SELECT * FROM users WHERE email = ?");
        $result = $statement->execute([$user->getEmail()]);
    }
    else
        return GENERAL_ERROR;
    if(!$result)
        return DB_ERROR;
    else
    {
        $result = $statement->fetch();
        if($result === false)
            return WRONG_EMAIL_OR_PASS;
        else
        {
            $pass = base64_encode(hash('sha256', $user->getPassword(), true));
            if (password_verify($pass, $result['password']))
            {
                $user->setPassword('');
                $user->setUserID  ($result['IDUser']);
                $user->setEmail   ($result['email']);
                $user->setUsername($result['username']);
                $user->setName    ($result['name']);
                $user->setSurname ($result['surname']);
                $user->setAddress ($result['address']);
                $user->setBirth   ($result['birthdate']);
                $user->setVerified($result['verified']);
                $user->setPrivilege($result['privilege']);
                return SUCCESS;
            }
            else
                return WRONG_EMAIL_OR_PASS;
```

3.3. Sistema di User Management (Gestione utenti)

Il sistema di User Management (gestione dell'utente) è situato nel Backend, include tutte le classi, i metodi e le variabili che riguardano l'utente

Il sistema è composto principalmente da due classi

- **User**
  contiene i dati dell'utente e i metodi a servizio dell'utente, esegue dei controlli sulle variabili mandate dall'utente durante l'accesso o la registrazione, chiama i metodi di DbManager solo quando le variabili passate hanno superato tutti i controlli.
- **DbManager**
  Trattata nel capitolo precedente

Quando un visitatore richiede l'accesso o la registrazione,
action.php crea un istanza di **User** e passa le credenziali dell'utente (mandate in POST) al metodo di User "login(...)" o "register(...)".

```
action.php

	$user = new User();
	$result = $user->login($emailusername, $password);
```

Le funzioni "login(...)" e "register(...)" eseguono ulteriori controlli sui dati passati dall'utente (per una questione di sicurezza) utilizzando le espressioni regolari.
Le espressioni regolari, o **RegEx** (da **Reg**ular **Ex**pression) sono
sequenze di caratteri che definiscono un modello di ricerca.

RegEx di esempio:

```
constants.php
	const REGEX_PASS    = '/^[a-zA-Z0-9!@#$%^&*_.~-]+$/';
user.php
	$result = preg_match(REGEX_PASS, $tmp_passwd);
	if(!$result)
		return INVALID_PASSWORD;
```

Se le stringhe passate dall'utente contengono dei caratteri non permessi, la funzione manderà dei messaggi di errore definiti nel file **constants.php**

Nel caso in cui le variabili superano tutti i controlli, User chiamerà il metodo login della classe statica **DbManager** (che gestisce il database) eseguendo la query per cercare l'utente in sicurezza.

user.php
```
$result = DbManager::query_login($this);

if($result === SUCCESS)
{
        $this->setLoginStatus(true);
}
return $result;
```

Se l'utente riesce ad eseguire l'accesso o la registrazione, l'istanza di User viene salvata all'interno della sessione del server serializzando i dati, (convertendo l'istanza in stringa memorizzabile).

action.php
```
if($result == SUCCESS) // LOGIN SUCCESSFUL
{
        session_regenerate_id(true);
        $user->setSessionID(session_id());
        $_SESSION['user_cache'] = serialize($user);
        $_SESSION['attempts'] = 0;
}
```

Così facendo si possono ricavare i dati dell'utente quando il server riceve nuove richieste e diminuisce l'overhead del database.

action.php
```
/* ------- USERS CACHE ------- */
$user = null;
if(isset($_SESSION['user_cache']))
{
    $user = unserialize($_SESSION['user_cache']);
}
```

3.4. Gestione e sicurezza delle sessioni

In PHP, le sessioni offrono un sistema sicuro dove memorizzare i dati di un utente connesso al server, la sessione scade automaticamente dopo 1440 secondi (24 minuti) di inattività.

In questo progetto ho implementato un sistema per far scadere la sessione dopo 10 minuti di inattività, aumentando la sicurezza nel caso che l'utente acceda da luoghi pubblici.

La sessione viene identificata tramite un "**SESSION ID**" che può essere personalizzato dal browser dell'utente, è solo grazie a questo che alcune funzionalità operano, come il "Restore Previous Session" che permette di ripristinare tutte le sessioni precedenti alla chiusura del browser.
Questa funzionalità comporta anche un rischio, espone l'utente a due tipi di attacchi: "Session fixation" e "Session hijacking" che consistono nel rubare o imporre il SESSION ID che può essere utilizzato per fingersi l'utente originale.

Questi tipi di attacchi si possono prevenire con diverse tecniche, sia a livello di client che di server.
In quanto al server, utilizzando **TLS** previene già attacchi di tipo sniffing come **MITM** (Man in the middle) quindi il Session ID **non** si può ricavare tramite la comunicazione client-server.
Questo non protegge il client da un attacco spyware, per questo motivo dovrebbe avere almeno un software antivirus.
Come visto nel codice all'interno del capitolo precedente, il server rigenera il SESSION ID garantendo una maggiore sicurezza una volta che l'utente viene autenticato.

Un altra sicurezza è il “**Session Timeout**”
Lo script sottostante ne dimostra il funzionamento, si richiede al server l’ora corrente e la confronta con **$SESSION[‘LAST_ACTIVITY’]** mentre $timeout_duration indica il tempo di timeout.

$SESSION[‘LAST_ACTIVITY’] si resetta ad ogni azione dell’utente.

```php
/*------- SESSION TIMEOUT CHECK -------*/
    $session_time = $_SERVER['REQUEST_TIME'];
    $timeout_duration = 600; //10 minute session timeout

    if (isset($_SESSION['LAST_ACTIVITY']) &&
    ($session_time - $_SESSION['LAST_ACTIVITY']) > $timeout_duration)
    {
        if($user != null)
        {
            $user->logout();
            $user = null;
            session_unset();
            session_destroy();
            session_start();
        }
    }
    //TIME RESETS ON EVERY ACTION
    $_SESSION['LAST_ACTIVITY'] = $session_time;
```

3.5. Prevenzione di attacchi informatici DoS e Bruteforce

Quando un sito web tratta i dati sensibili dei propri utenti, la sicurezza è una delle maggiori preoccupazioni.
Iniziamo dalla trasmissione dei dati dal client al server, normalmente la connessione avviene tramite il protocollo non criptato HTTP che manda i dati in chiaro (quindi sono leggibili ed intercettabili).

L'unico modo sicuro per mandare i dati assicurandosi che non siano modificabili o intercettabili è stabilire una connessione con il protocollo HTTPS, che per funzionare richede un certificato SSL.
Per nostra fortuna, Il certificato è comunemente pre-installato nel server dall'hosting provider.
Il protocollo HTTPS stabilisce una connessione sicura e criptata tra il client e il server grazie al protocollo **TLS** (Transport Layer Security).

In questo capitolo analizzeremo alcune vulnerabilità con le corrispondenti soluzioni pratiche.

ReDoS (Regular Expression Denial of Service)

Gli attacchi DoS classici non sono più efficaci,
L'hosting provider e PHP impediscono gli attacchi Denial of Service tramite diverse pratiche, ma non rendono il server invulnerabile ad un attacco mirato.

Le espressioni regolari sono CPU-Intensive, hanno un tempo di esecuzione esponenziale direttamente proporzionale alla lunghezza della stringa e del RegEx.
Questa caratteristica può essere sfruttata da un malintenzionato per condurre un attacco DoS mirato mandando ripetute richieste di login o registrazione stile **Bruteforce**.
L'attacco può avere anche un secondo fine, quello di trovare degli account vulnerabili.

Come si può prevenire l'attacco ReDoS e Bruteforce?
Esiste una soluzione semplice ed efficace ad entrambi i problemi.
Si tratta di uno script che stabilisce un tempo di attesa per eseguire la richiesta dopo aver tentato di accedere più di 5 volte

```
   /*------- ReDoS PROTECTION -------*/
    $time_wait = 0; //Esponenziale

    if(isset($_SESSION['attempts']))
    {
        $attempts = $_SESSION['attempts'];
        if($attempts > 5)
            $time_wait = $attempts / 2;
    }

    if (isset($_SESSION['LAST_ACTIVITY']) &&
    ($_SERVER['REQUEST_TIME'] - $_SESSION['LAST_ACTIVITY']) < $time_wait)

    {
        echo TOO_MANY_REQUESTS;
        exit();
    }

$_SESSION['LAST_ACTIVITY'] = $_SERVER['REQUEST_TIME'];
```

"attempts" sta per "tentativi" ad ogni tentativo di login o registrazione **$attempts** incrementa di uno mentre **$time_wait** corrisponde al tempo di attesa in secondi, se time_wait è maggiore di o verrà comunicato l'errore TOO_MANY_ATTEMPTS visualizzato come notifica al client.

```
   /* ------- LOGIN FORM ------- */
    if(isset($_GET['submit_login']))
    {
        if(isset($_SESSION['attempts']))
            $_SESSION['attempts'] = $_SESSION['attempts'] + 1;
        else
            $_SESSION['attempts'] = 1;
```

La variabile $attempts viene salvata a livello di sessione, e viene resettata nel momento che l'utente esegue il login.

3.6. Una soluzione per la visualizzazione mobile

Durante lo sviluppo della parte grafica, la prima cosa che mi sono chiesto è stata: come posso presentare il sito in versione mobile?

Così mi è tornato in mente Wordpress, durante la creazione del sito
di mio padre avevo utilizzato un sistema che si chiamava
WP-Bakery, permetteva di creare pagine modularmente e gestiva tutto tramite righe e colonne.
Visualizzando il sito da cellulare, le colonne venivano automaticamente convertite in righe.
Così mi è venuta l'idea di creare un sistema simile a quello di Wordpress, trasformando le colonne in righe tramite il Frontend.

La classe **ResponsiveManager** mette in pratica questo concetto.
Durante l'inizializzazione del Frontend la classe controlla tramite una macro se il browser utilizzato è in modalità mobile o se ha una risoluzione troppo piccola per la visualizzazione desktop.

Se scatta la modalità mobile le classi degli elementi a colonna vengono rimosse e sostituite con uno spaziatore.

Naturalmente lo script controlla per cambiamenti nella dimensione del browser, così da ritornare alla modalità desktop una volta che la risoluzione supera la soglia di 800 pixel (impostata dal parametro del costruttore chiamato nella classe UIManager)

responsive.ts

```ts
//toggle between mobile and desktop mode
private mobileMode( status: boolean )
{
  // MOBILE MODE
  if (status === true && status !== this.mobileStatus)
  {
    if (
      this.allColumnsElements != undefined &&
      this.allRowElements != undefined &&
      this.shadowElements != undefined &&
      this.modals != undefined
    ) {
      for (let index = 0; index < this.allColumnsElements?.length; index++) {
        this.allColumnsElements
          .item(index)
          .classList.remove(
            'div_internal_column',
            'column_width_1e6',
            'column_width_1e5',
            'column_width_1e4',
            'column_width_1e3'
          );
        this.allColumnsElements.item(index).classList.add('paddingBottom20');
      }
      //ADD flexColumns TO Rows
      for (let index = 0; index < this.allRowElements?.length; index++) {
        this.allRowElements.item(index).classList.add('flexColumn');
      }
      //REMOVE boxshadow FROM shadowElements
      for (let index = 0; index < this.shadowElements?.length; index++) {
        this.shadowElements.item(index).classList.remove('boxshadow');
      }
      //ADD modal_content_mobile TO modals
      for (let index = 0; index < this.modals?.length; index++) {
        this.modals.item(index).classList.add('modal_content_mobile');
      }
    }
    this.mobileStatus = status;
  }
```

3.7. Il funzionamento della dashboard (AJAX e XML)

Le richieste **AJAX (Asynchronous JavaScript and XML)** possono inviare e recuperare dati dal server in modo asincrono (in background) senza interferire con la visualizzazione e il comportamento della pagina.

Pillola di storia:
Dal 2017 a JavaScript è stato aggiunto il **Fetch API**, un sistema più flessibile che punta a sostituire AJAX, in questi ultimi anni i browser si stanno aggiornando per supportarlo.

In questo progetto le richieste AJAX sono state utilizzate per tutte le funzionalità del sito, per questo motivo ho inventato un sistema semplice per gestire tanti tipi di richieste.

ho creato una classe per gestire le richieste chiamata **AjaxManager**
contiene pochi metodi, il più importante è questo:

ajaxmanager.ts

```
    /**
     * For all other requests
     * @param callback function to be called after server response
     */
    public ajax_custom_request(param: string, callback: Function)
    {
        let xmlRequest = new XMLHttpRequest();
        let url = 'user_management/action.php';

        xmlRequest.addEventListener('load', () => { callback(xmlRequest); } );

        xmlRequest.open('POST', url, true);
        xmlRequest.setRequestHeader('Content-type',
                                    'application/x-www-form-urlencoded');
        xmlRequest.send(param);
    }
```

Il parametro **callback** è una funzione che viene richiamata una volta che arriva il risultato dal server.
Nel parametro **param** viene inserita una costante che definisce la richiesta definita nella classe **Constants** per semplificarne l'utilizzo

constants.ts

```
    /*USER*/
public static readonly REQUEST_MAP_MARKERS      = 'get_map_data=true';
public static readonly REQUEST_LOGIN_STATUS     = 'check_user=true';
public static readonly CHECK_PRIVILEGE_STATUS   = 'check_user_privilege=true';
    /*DASHBOARD */
public static readonly REQUEST_OVERVIEW         = 'dashboard=true&overview=true';
public static readonly REQUEST_ACCOUNT_DETAILS  = 'dashboard=true&account-details=true';
public static readonly REQUEST_BILLING_DETAILS  = 'dashboard=true&billing-details=true';
public static readonly REQUEST_ACTIVITY_DETAILS = 'dashboard=true&activity-details=true';
public static readonly REQUEST_ADMIN_MANAGEMENT = 'dashboard=true&admin-management=true';
public static readonly REQUEST_ADMIN_MANAGE_STATIONS = 'dashboard=true&manage-stations=true';
public static readonly REQUEST_ADMIN_MANAGE_BIKES    = 'dashboard=true&manage-bikes=true';
public static readonly REQUEST_ADMIN_MANAGE_USERS    = 'dashboard=true&manage-users=true';
    /*DASHBOARD QUERIES */
public static readonly REQUEST_QUERY_ACTIVITY        = 'dashboard=true&query-activity=';
public static readonly REQUEST_ADMIN_QUERY_STATIONS  = 'dashboard=true&query-stations=';
public static readonly REQUEST_ADMIN_QUERY_BIKES     = 'dashboard=true&query-bikes=';
public static readonly REQUEST_ADMIN_QUERY_USERS     = 'dashboard=true&query-users=';
```

La classe **UserHandler** effettua il login, il logout,la registrazione, e il controllo dello stato di autenticazione tramite questa funzione.

Mentre la classe **DashManager** la utilizza per visualizzare i dati e le attività dell'utente, e comprende anche le funzionalità per il pannello admin.

Prendo come esempio la pagina “Attività” nella bacheca perchè è un ottimo esempio per spiegare come funzionano le richieste Ajax e gli XML.
La pagina attività visualizza una tabella contenente le ultime attività dell’utente che si può scorrere pagina per pagina.

dashboard.ts

```
        this.buttActivity?.addEventListener('click', () => {

          this.handlePageBrowsing(Constants.REQUEST_ACTIVITY_DETAILS, (x: number) =>
          {
            this.handleActivities(x);
          })
        });
```

la richiesta della pagina attività è divisa in due richieste, effettuate dalla funzione **handlePageBrowsing** che gestisce le pagine con tabelle.
La prima richiesta chiederà al server di mandare la pagina “**activity.php**”

Al suo interno ci sarà l'intestazione della pagina e della tabella che verrà visualizzata all'interno di un contenitore sottostante al menù della bacheca.

la seconda richiesta è gestita dalla funzione **handleActivities(x)** che verrà eseguita nel momento che **activity.php** sarà visualizzata.
il parametro **x** di **handleActivities** identifica il numero della pagina richiesta.
Il server manderà in risposta le attività dell'utente (ordinate dalla più recente) in formato XML per costruire la tabella.

XML (Extensible Markup Language)

è un linguaggio di markup come HTML basato su una sintassi che consente di definire e controllare il contenuto degli elementi all'interno un documento. Nel nome è definito "Estensibile" perchè permette di creare **tag** personalizzati.

Il punto forte del formato XML è la sua semplicità, dalla creazione alla lettura.
È uno dei formati più utilizzati per la trasmissione dati tramite internet, può essere utilizzato fondamentalmente da quasi tutti i linguaggi conosciuti ed è quindi un formato universale e personalizzabile.

Esempio di XML mandato dal server

```xml
<user_activities>
    <activity>
        <date>2022/01/26</date>
        <origin>Centrale</origin>
        <destination>Centrale</destination>
        <duration>01 Ore e 0 Min</duration>
        <distance>0.0</distance>
    </activity>
    <activity>
        <date>2021/10/13</date>
        <origin>Porta Nuova</origin>
        <destination>Centrale</destination>
        <duration>01 Ore e 0 Min</duration>
        <distance>0.7</distance>
    </activity>
</user_activities>
```

Il Frontend dovrà effettuare il **parsing** dell'XML per costruire la tabella.

Che cos'è il parsing ?
È un termine che si utilizza in informatica che significa dissezionare una struttura dati per ricavarne i singoli dati.

Durante il parsing i singoli dati vengono utilizzati per creare la tabella finale che viene visualizzata.

4. Matematica

4.1. Formula di Haversine

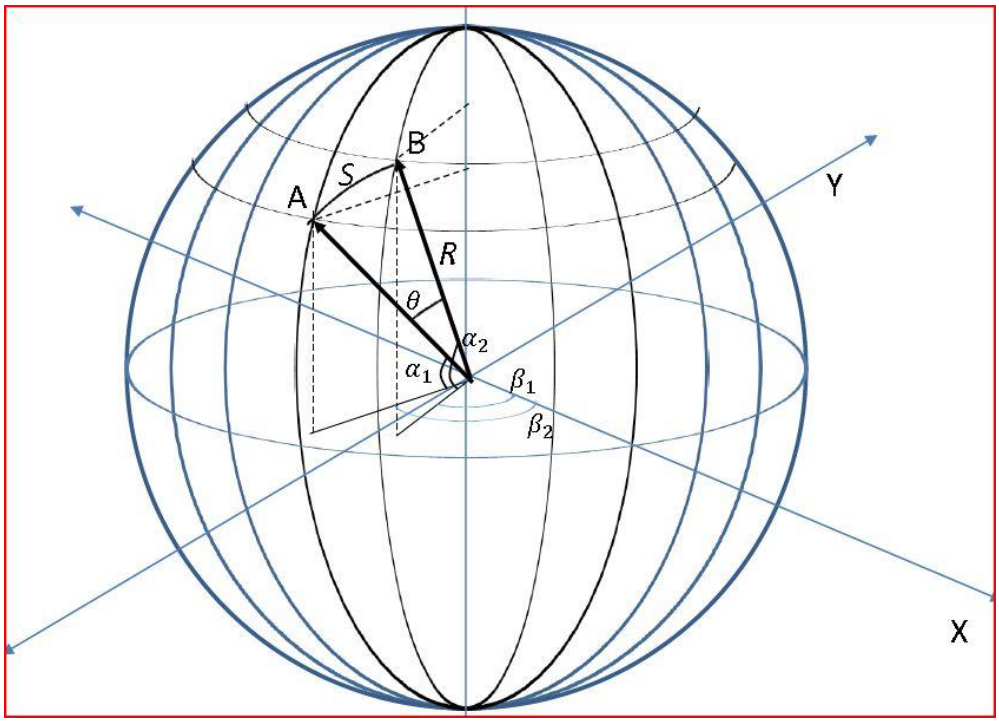
la formula "**Haversine**" determina la distanza grande cerchio tra due punti su una sfera date le longitudini e latitudini . Importante in navigazione , è un caso particolare di una formula più generale trigonometria sferica.

Nel progetto è utilizzata per calcolare la distanza tra due punti di latitudine e logitudine di stazioni diverse per stimare la distanza in km.

ϕ è latitudine
λ è longitudine
R è il raggio terrestre (raggio medio = 6.371 km)

$a = \sin^2(\Delta\phi / 2) + \cos\phi_1 \cdot \cos\phi_2 \cdot \sin^2(\Delta\lambda / 2)$
$c = 2 \cdot \text{atan2}(\sqrt{a}, \sqrt{(1 - a)})$
$d = R \cdot c$

Gli angoli sono espressi in radianti.



A è il quadrato della lunghezza di una corda tra i due punti.

C è la distanza angolare in radianti utilizzata per calcolare la distanza in metri.

```
        lat1 = radians(stationsLat[stationOrigin-1])
        lon1 = radians(stationsLon[stationOrigin-1])
        lat2 = radians(stationsLat[stationDest-1])
        lon2 = radians(stationsLon[stationDest-1])

        dlon = lon2 - lon1
        dlat = lat2 - lat1

        a = sin(dlat / 2)**2 + cos(lat1) * cos(lat2) * sin(dlon / 2)**2

        c = 2 * atan2(sqrt(a), sqrt(1 - a))

        distance = "%.1f" % (R * c)
```

5. Sviluppi Futuri

Non ho avuto il tempo di sviluppare tutte le funzionalità che avrei voluto implementare, a partire dal sistema di Email SMTP gestito tramite PHPMailer.

PHPMailer è una libreria che gestisce la connessione tramite il server SMTP e rende possibile mandare email complesse, con allegati e formattazione, è utilizzata anche in Wordpress.

Tramite il sistema di email si possono sviluppare le funzionalità relative all'utente, come le funzioni "Password dimenticata" e "Verifica account".

Ho predisposto il database per aggiungere la funzionalità "Ricordami" che funziona tramite dei token generati in modo casuale e salvati client-side in cookie HTTP-ONLY (accessibili solo dal server e non da JavaScript).

Sarebbe molto bello poter migliorate questo progetto utilizzando anche un Framework come Vue.JS.

Lo sviluppo web che è insegnato a scuola è ormai un linguaggio obsoleto superato da queste nuove tecnologie.

Questo progetto potrebbe essere un valido inizio da sviluppare in futuro con studenti interessati a portarlo a compimento.
Difatti ho riunito tutto il materiale necessario per riprendere il lavoro su GitHub, per chi fosse interessato a proseguire il mio lavoro.

Sono disponibile per chiunque voglia creare il proprio branch su GitHub in modo da continuare lo sviluppo del sito.

Sarebbe un piacere per me sapere che questo progetto potrà essere portato a termine..

6. Commento

Berghem Bike è stato un progetto che ha incoraggiato la mia creatività scoprendo il mondo dello sviluppo web che ho trovato molto coinvolgente.

Nel corso dello sviluppo del sito ho affrontato imprevisti e problematiche che mi hanno permesso di migliorarmi e scoprire che è un settore che mi interessa molto.

Ho inserito il mio elaborato nel mio curriculum come candidato all'ITS JobsAcademy nel settore del digital marketing rendendo possibile la mia ammissione dopo aver affrontato un esame molto impegnativo che ho superato con ottimi voti.

Inoltre, ho collaborato recentemente con un'azienda di Milano la Digital Content Creators per risolvere dei problemi tecnici nel sito di mio padre, ho avuto il piacere di conoscere il titolare Gianluca Colombi che da vent'anni lavora nel settore, è rimasto molto colpito dal mio lavoro e dalla mia intraprendenza.

Mi è stato offerto un lavoro per quest' estate, difatti sarò affiancato da un suo tecnico per collaborare ad alcuni loro progetti.

Questo lavoro mi permetterà di migliorare la mia esperienza e toccare con mano questo settore lavorativo che m'interessa molto.

7. Glossario

Client

fa riferimento al browser dell'utente che si connette al server

Server

fa riferimento al servizio all'interno di un computer che contiene il sito web, rendendolo accessibile all'internet tramite un indirizzo specifico.

Frontend

La rappresentazione visiva del sito web, è composta da codice HTML (Markup Language che definisce gli elementi della pagina), CSS (Foglio di stile), e JavaScript (Linguaggio di programmazione)

Backend

è la parte contenuta nel server che gestisce l'accesso ai dati, sito web e database.

Framework

è un sistema che fornisce funzionalità generiche, può essere selettivamente modificato da un codice scritto aggiuntivo dell'utente, fornendo così un software specifico per l'applicazione.

Classe

è un contenitore che contiene metodi e le variabili, viene chiamato oggetto nel momento in cui viene creata un istanza di esso in memoria.

Istanza

L'istanza è un oggetto inizializzato e in memoria.

Metodo

Un metodo equivale ad una funzione, si chiama metodo dal momento che è all'interno di una classe.

Funzione

Una funzione è un pezzo di codice che viene chiamato attraverso un nome, la funzione può avere anche dei parametri che vengono passati nella chiamata.

Stringa

È una sequenza di caratteri.

Sessione del server

In termini informatici è una cache (un magazzino temporaneo) dove vengono salvate informazioni temporanee all'interno di un file collocato nel server.
Le informazioni sono accessibili al Backend anche se l'utente cambia pagina.
La sessione è relativa alla specifica connessione, finchè il browser del client rimane aperto la sessione rimane attiva.
Se passano 30 minuti dall'ultima richiesta la sessione viene automaticamente eliminata. Con il sistema di sicurezza aggiunto da me dopo 10 minuti la sessione viene resettata.

Sessione del browser

In termini informatici è una cache (un magazzino temporaneo) dove vengono salvate informazioni temporanee nel browser dell'utente a cui il Frontend può accedervi anche se l'utente cambia pagina.

GET.. POST

Sono due tipi di richieste del protocollo HTTP che possono essere effettuate dal client, entrambe le richieste possono contenere dei parametri, quando bisogna mandare dei dati sensibili il metodo POST offre una maggiore sicurezza. Il metodo GET è utilizzato principalmente per richiedere risorse specifiche, come richiedere una pagina in lingua.

Diminuire l'overhead

Diminuire l'overhead significa diminuire il numero di query richieste per effettuare le stesse operazioni, come visualizzare i dati dell'utente.

Hosting Provider

È un azienda che offre uno spazio web ad un costo mensile o annuale.

Query

È una richiesta effettuata dal server al database

8. Referenze

Repository Berghem Bike su GitHub
https://github.com/PrometeoPrime/Berghem_Bike

Vulnerabilità e soluzioni per sviluppatori
https://rules.sonarsource.com/php

OWASP Authentication cheat sheet
https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html

Guida concettuale per un sistema di autenticazzione sicuro
https://stackoverflow.com/questions/549/the-definitive-guide-to-form-based-website-authentication#477579

Guida da un esperto di stack overflow sul PDO
https://phpdelusions.net/pdo#why

Formula di Haversine e applicazioni
https://www.movable-type.co.uk/scripts/latlong.html