

GESTIÓ HOTELERA

Hotel Trampolín

Hotel Trampolín

Iniciar sessió

Home

Hotel

Habitacions

Contacte

Arrivada

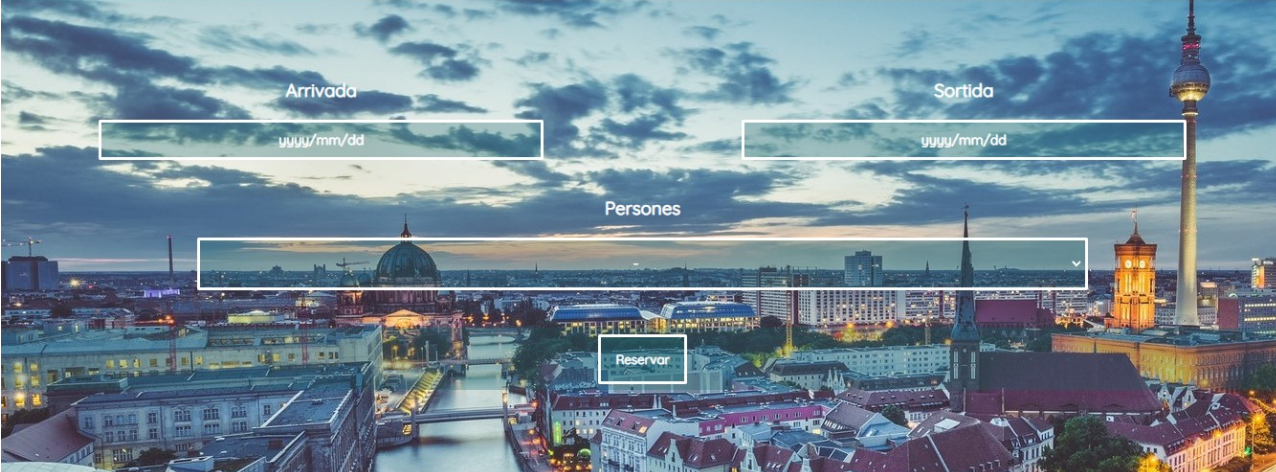
yyyy/mm/dd

Sortida

yyyy/mm/dd

Persones

Reservar



Índex

Planificació.....	3
Servidor Debian.....	3
IDE.....	3
Git i Github.....	3
Wireframe.....	4
Pàgina web.....	4
Requisits.....	5
Tests de funcionament.....	5
Enllaçar amb la base de dades.....	5
Comprovacions de les consultes a la base de dades.....	6
Comprovar enviament de dades a través de formularis.....	7
Comprovar el correcte funcionament dels CRUD.....	7
Dates de reserva.....	7
Casos d'ús.....	8
Usuaris.....	8
Procés de reserva.....	9
CRUDS.....	9
Model Entitat/Relació.....	10
Taula rol.....	10
Taula loginDB.....	10
Taula client.....	10
Taula reserva_client.....	10
Taula reserves.....	10
Taula tipoHabitacio.....	10
Taula habitacio.....	10
Taula contacte.....	10
Milliores.....	11
Enviar correu amb les dades de la reserva.....	11
Filtrar llista de reserves.....	11
Pàgina "Hotel".....	11
Enllaç Github.....	12

Planificació

Servidor Debian

Per començar a fer la nostre web, el primer és instal·lar el servidor php que, nosaltres ja el teníem instal·lat i configurat correctament de l'anterior projecte però aquest cop li hem afegit també una interfície de xarxa d'adaptador pont per a que el company de desenvolupament del projecte pugui tenir accés al servidor.

IDE

Per a anar creant i editant de forma més còmode farem servir un IDE des de la nostre màquina real i transferirem els fitxers a la màquina servidor cada cop que vulguem aplicar els canvis a la nostre pàgina amb una comanda del security copy protocol.

En el nostre cas farem servir Visual Studio Code ja que ens permet instal·lar moltes extensions que ens poden facilitar o ajudar molt a optimitzar el nostre temps desenvolupant la web i a trobar fallos.

Algunes de les extensions que hem instal·lat són les següents:

- **Auto Rename Tag**
 - Canvia el nom automàticament d'una etiqueta HTML/XML aparellada.
- **Live Server**
 - Veure de forma dinàmica una pàgina html.
- **PHP Debug**
 - Per a debugar el codi PHP i trobar errors.

Nota: Aquesta última s'ha de configurar un cop instal·lada per a poder-la utilitzar.

Git i Github

Hem creat cadascú un **repositori local** on serà el nostre entorn de treball personal que posteriorment **vincularem** amb un nou **repositori del Github** per a poder **compartir els canvis** que vagi fent cada un així els meus canvis es reflectiran en els fitxers del meu company i al revés.

Wireframe

Un cop hem instal·lat les eines per a poder començar a treballar amb la web, necessitarem saber més o menys quina estructura tindrà.

Per això hem fet un wireframe de les pàgines que creiem que és més important saber-ne l'estructura com per exemple: la pàgina d'inici, la pàgina que mostrarà les habitacions disponibles, la pàgina on l'usuari veurà les dades de l'habitació abans de fer la reserva i la pàgina amb el formulari de reserva.

Domini

Per a crear un nou domini al servidor de proves vam seguir el següent [tutorial](#).

El qual ens indica que hem de crear unes carpetes i posteriorment editar un fitxer de sites-available per a definir el domini i a quines carpetes farà referència.

Pàgina web

Un cop tenim ja una base començarem a fer la web, primer de tot farem l'estructura bàsica de cada pàgina i un cop acabada passem a estilitzar-la. Per fer la pàgina més amigable per l'usuari hem utilitzat llibreries de **jquery** per fer coses com per exemple **sliders d'imatges o lightbox**.

Utilitzarem el patró **model, vista, controlador** per a l'esquelet de la pàgina, d'aquesta forma, crearem dues carpetes una amb el nom «**public**» que contindrà l'index.php, els estils, imatges i js. Farem una altra carpeta amb el nom «**src**» la qual tindrà més carpetes dins seu: **middleware, controladors, vistes, models, emeset**. També hi haurà el fitxer principal de configuració, el config.php

Cada carpeta d'aquestes guardarà fitxers que tindran diferents funcions.

Per fer gestors tant d'usuaris, com d'habitacions, com de reserves farem un **CRUD (Create Read Update Delete)** a una pàgina diferent per cada cosa. A aquestes pàgines només hi podran accedir alguns usuaris. Això ho controlarem amb el middleware.

Requisits

- Per a crear el projecte del joc de la vida necessitarem un **servidor amb PHP** on hi desarem tots els fitxers relacionats amb l'àmbit web com per exemple els **html, php, js, css**, etc...
- També necessitarem un **IDE** per a poder modificar i afegir codi als fitxers que es troben dins el servidor, en el meu cas he fet servir Visual Studio Code.
- Hem d'instal·lar el **Git** per a poder vincular el repositori local amb el repositori del **github** per a poder anar desant els canvis al núvol i no perdre'ls en cas de modificació errònia o borrat dels fitxers i també per a compartir el projecte amb el company.
- Necessitarem crear una base de dades la qual emmagatzemarà totes les dades referents als clients, diferents tipus d'usuaris que hi pugui haver a la web, les reserves i les habitacions.
- Farem una **connexió ftp** per a enviar els diferents fitxers de la web que anem desenvolupant, cap al servidor i així poder reflectir els canvis a la pàgina.

Tests de funcionament

Enllaçar amb la base de dades

Els primers tests que hem hagut de fer van ser per comprovar la connexió amb la base de dades i veure que està tot vinculat correctament. Per fer això de forma ràpida, simplement hem posat un condicional if de la següent manera:

```
if ($conex)
{
    echo "Connexió amb la base de deades establerta correctament";
    die();
}
```

Un cop hem comprovat que enllaça correctament amb sqli, hem passat a PDO fent servir les mateixes dades per entrar a la base de dades.

Comprovacions de les consultes a la base de dades

Comencem a tractar amb les dades de la base de dades, per veure que les consultes funcionen correctament i ens retorna el que volem, primer de tot hem fet els consultes des d'un terminal mariadb:

```
MariaDB [gestio_hotelera]> SELECT * FROM loginDB WHERE usuari = 'alex';
+-----+-----+-----+-----+
| id | usuari | contrasenya | rol |
+-----+-----+-----+-----+
| 1 | alex | 1234 | 1 |
+-----+-----+-----+-----+
1 row in set (0.036 sec)
```

Posteriorment això s'ha de adaptar a PDO per a poder fer la consulta des de la nostre pàgina web, com que hi ha un camp variable en la nostre consulta haurem de fer servir estaments preparats de PDO, llavors la consulta ens quedaria en forma de funció de la següent manera:

```
public function getUser($user)
{
    $query = 'select id, usuari, contrasenya, rol from loginDB where usuari=:user;';
    $stm = $this->sql->prepare($query);
    $result = $stm->execute([':user' => $user]);

    if ($stm->errorCode() !== '00000') {
        $err = $stm->errorInfo();
        $code = $stm->errorCode();
        die("Error.  {$err[0]} - {$err[1]}\n{$err[2]} $query");
    }

    return $stm->fetch(\PDO::FETCH_ASSOC);
}
```

Finalment, per fer les proves de que ens retorna el que volem també des de la web, hem fet servir la comanda **var_dump**, **echo** i **print_r**.

```
$usuaris = new \Daw\UsuarisPDO($contenedor->config["db"]);
// CRIDA DE FUNCIONS PER AGAFAR LES DADES DELS USUARIS
$llistaUsuaris = $usuaris->selectAll();

print_r($llistaUsuaris);
die();
```

Això ho fem amb tots els casos que vulguem veure què ens retorna una consulta a la base de dades, fer servir **var_dump** és la millor opció en tots els casos però també podem fer servir **print_r** per veure contingut d'arrays i **echo** per veure strings, integers, etc...

Comprovar enviament de dades a través de formularis

Per a comprovar que les dades que s'emplenen en el formulari s'envien correctament per a ser tractades farem servir el mètode **GET** ja que amb aquest podem veure si s'envien les dades de forma bastant simple mirant la URL de la nostra pàgina i no ens caldrà fer un **echo** per cada dada que vulguem veure per pantalla. Un cop veiem que les dades s'envien correctament, ho passem tot a **POST** per a fer que aquestes dades no siguin visibles i per tant, no puguin ser manipulades per l'usuari.

No seguro | hoteltrampolin.com/public/index.php?r=editUser&id=1

Comprovar el correcte funcionament dels CRUD

Ja que un CRUD té diferents accions per fer, s'haurà de comprovar el correcte funcionament de cada una. Primer prepararem i provarem les consultes a la base de dades com hem vist anteriorment, un cop tinguem les consultes, haurem de incorporar-les a cada botó referent:



Per comprovar ara que tot funciona, simplement provarem a crear un nou registre i mirarem si l'ha inclòs a la base de dades, farem el mateix per la funció d'editar un registre i la d'eliminar-ne un: després de haver provat cada botó, mirarem si els canvis s'han reflectit a la base de dades o al mateix CRUD, ja que agafa la informació de la mateixa taula de la base de dades.

#	Usuari	Contrasenya	Rol	
1	alex	1234	1	 
2	simon	1234	1	 

Dades de reserva

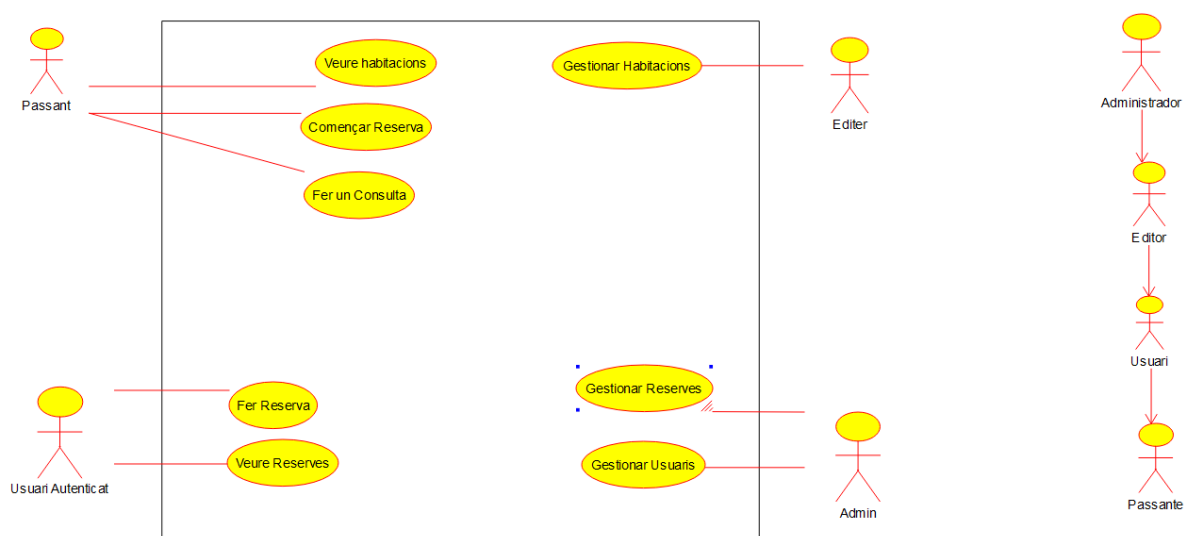
Per provar que les dades de reserva s'estaven creant de forma correcta amb el DateTime hem fet servir el **debugger de PHP** per veure com es van tractant les dades a mesura que va avançant el bucle del motor de reserves, d'aquesta forma, hem pogut veure si la data anava avançant dia a dia, la reacció de com guarda un tipus d'habitació en cas de estar disponible, etc etc...

També hem fet servir aquesta eina per fer proves amb altres dades com per exemple en algunes pàgines on hem de generar elements de forma dinàmica.

Casos d'ús

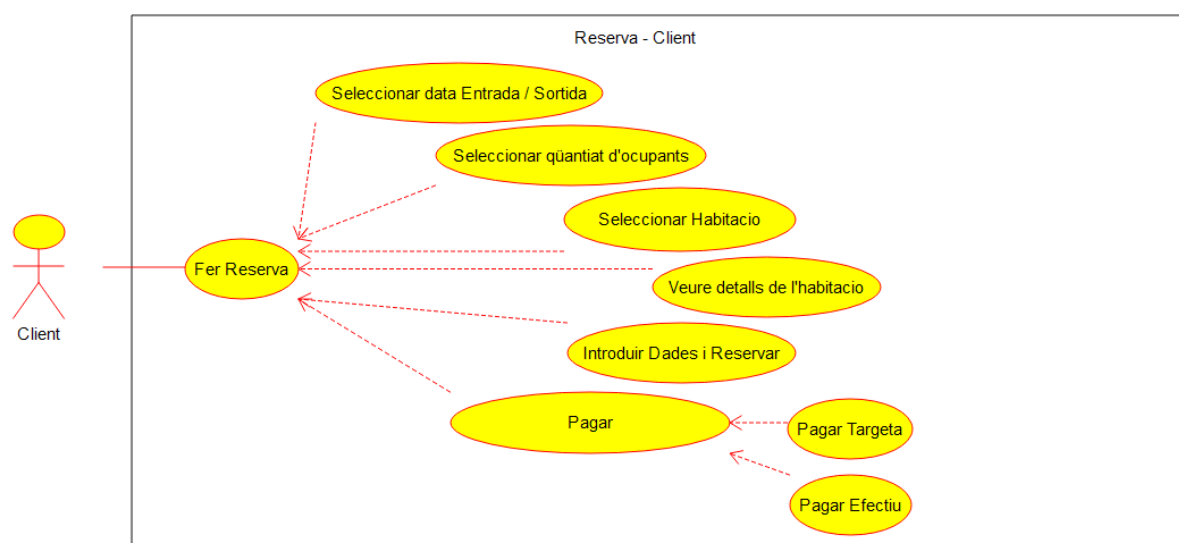
Per a facilitar la informació sobre **com es comunica i comporta el nostre sistema**, hem realitzat uns diagrames de casos d'ús del que hem considerat que és més important tenir en compte com per exemple: **els usuaris, quan el client fa una reserva i els diferents CRUD que hem fet.**

Usuaris



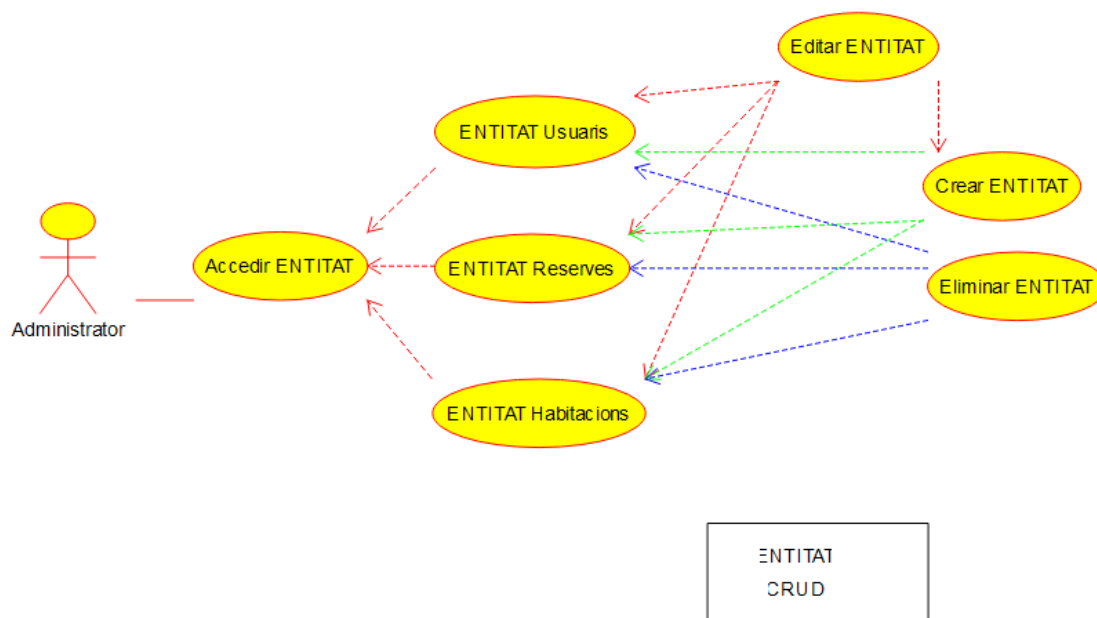
Aquí es poden veure els diferents tipus d'usuaris i quines accions poden realitzar cada un.

Procés de reserva



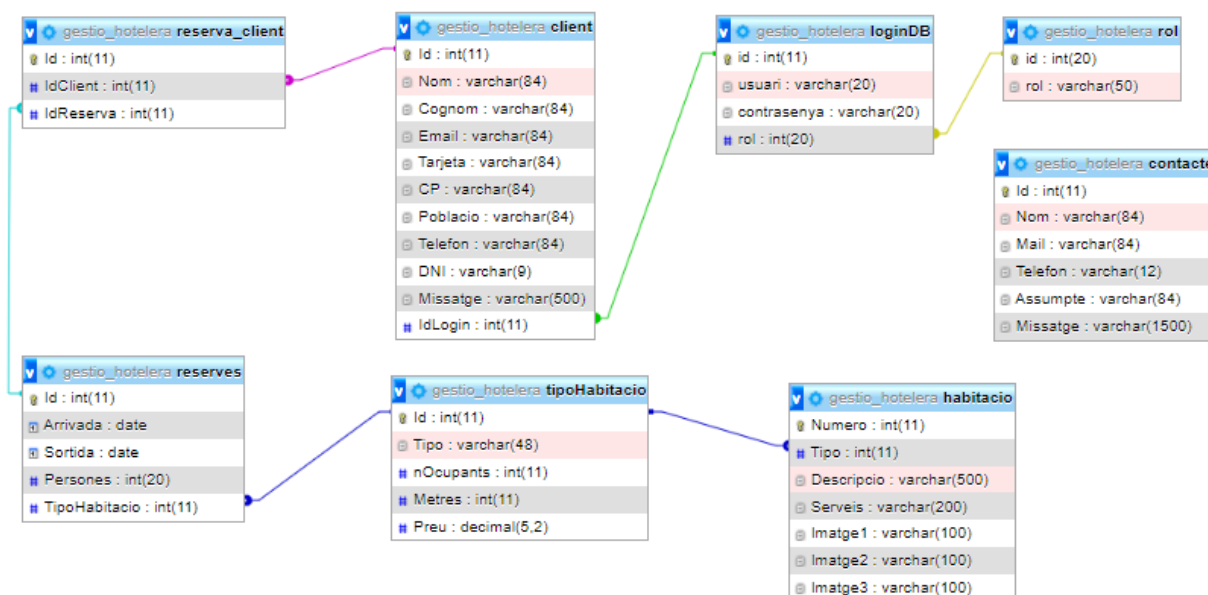
Podem veure quins passos ha de seguir l'usuari per a realitzar una reserva.

CRUDS



Aquí podem veure quines accions permeten fer els diferents CRUDS que tenim.

Model Entitat/Relació



Tenim un total de 8 taules a la nostra base de dades:

Taula rol

Conté els diferents tipus de rol que pot tenir un usuari.

Taula loginDB

Conté les dades de cada usuari per a iniciar sessió i també a quin rol pertany.

Taula client

Conté la informació personal del client que ha realitzat la reserva.

Taula reserva_client

S'utilitza per vincular un client amb una reservat.

Taula reserves

Guarda totes les reserves fetes fins al moment.

Taula tipoHabitacio

Té les dades de cada tipus d'habitació: el preu, la gent que hi cap i els metres quadrats.

Taula habitacio

Aquesta taula té les dades totes les habitacions del hotel de forma individual.

Taula contacte

Aquí guardem les dades del contingut del formulari de contacte enviat per qualsevol persona, client o no. Per això no està vinculada a cap altra taula.

Millores

Enviar correu amb les dades de la reserva

Com a punt extra a la nostre web, podem fer que **s'envii un correu al correu electrònic escrit al formulari** de reserva amb les dades de la mateixa, d'aquesta forma l'usuari podria consultar la seva reserva a través del correu o bé des de la mateixa pàgina a l'apartat de «Veure reserva».

Filtrar llista de reserves

Per a fer més fàcil trobar una reserva des de la gestió, podem **incorporar filtres** com per exemple buscar quines dates estan dins un **rang de dates** indicat, o mostrar quines dates són superiors a una altre o directament **filtrar per número de reserva**, nom de la persona que ha reservat o usuari que ha fet la reserva.

Per fer tot això ens ha faltat temps, però hagués sigut un molt bon punt a fer ja que facilitaríem molt el procés de trobar una reserva per un treballador.

Pàgina "Hotel"

Aquesta pàgina sempre ha estat en manteniment ja que hem dedicat la major part del temps a altres pàgines que depenen de backend per a tindre un correcte funcionament, tractament de dades, etc...

Aquesta pàgina contindria una **descripció de l'hotel i de l'equip de treballadors que hi ha per fer-lo funcionar**.

Enllaç Github

[Repositori github](#)