# Backend

Il backend è composto dall'insieme del codice php che gestisce le richieste dell'utente e interagisce con il database.

## Api

- ogni API ritorna una pagina HTML, che sia
  - get
  - post
  - errore
- un form errato ritorna una pagina con il form già compilato e un messaggio di erorre
- un form corretto reindirizza se è il caso
- per ogni API c'è una funzione o una classe che si occupa di gestire la richiesta e restituisce la pagina html.

## Database

Le connessione al database usa un Singleton del tipo:

```php
<?php
class Database {
  private static $instance = null;
  private $conn;
  private $host = 'localhost';
  private $user = 'username';
  private $pass = 'password';

  private function __construct() {
      Database::$conn = new mysqli($this->host, $this->user, $this->pass);
      // or something like that

      if (Database::$conn->connect_error) {
          die("Connection failed: " . Database::$conn->connect_error);
      }
  }

  public function getInstance() {
      if (Database::$instance == null) {
          Database::$instance = new Database();
      }
      return Database::$instance;
  }

  public query($sql_query, ...$params) {
      // check for sql injection
      // prepare statement
      // bind parameters
      // execute
      // return result
  }
}
```

## Autenticazione

Viene usata una classe `Session` per gestire la sessione dell'utente. Questa classe si occupa di gestire:

- la sessione

- l'autenticazione
- i cookies
- l'autorizzazione

E quindi implementa i metodi necessari per fornire queste funzionalità.

```php
<?php
class Session {
  public function start() {
      session_start();
  }

  public function set($key, $value) {
      $_SESSION[$key] = $value;
  }

  public function get($key) {
      return $_SESSION[$key];
  }

  public function destroy() {
      session_destroy();
  }

  public function is_logged_in() {
      return isset($_SESSION['user']);
  }

  public function login($user) {
      // check user credentials
      $_SESSION['user'] = $user;
  }

  public function logout() {
      unset($_SESSION['user']);
  }

  public function get_user() {
      return $_SESSION['user'];
  }

  ...
}
```

## Pagina

La generazione delle pagine html è gestita da una classe `Page` che si occupa di:

- leggere i file html
- sostituire i placeholder con i dati
- restituire la pagina html

```php
<?php
class Page {
  public function get_content($path) {
      $content = file_get_contents("layout/" . $path);
      $content = str_replace("{{ header }}", $this->get_header(), $content);
  }
```

```
  public function get_header() {
      return file_get_contents("layout/header.html");
  }

  // ...
}
```

## Componenti

Da `Page` si estendono le classi che generano l'html per ogni componente della pagina. Queste classi interagiscono con il db e generano l'html per ogni componente. Di seguito qualche esempio:

- `Recipe` che implementa i metodi:
  - `get_form()`
  - `get_item()`
  - `update(...$params)`
- `Ingredient` nella pagina `ingredient.php`
  - `get_form()`: non so bene come gestire questo
  - `get_item()`
  - `update(...$params)`
- `User` nella pagina `user.php`
  - `get_login_form()`
  - `get_register_form()`
  - `get_update_form()`
  - `get_item()`: just to display user info
  - `update(...$params)`

## Test

Fare una utility per eseguire i test. Questa utility estende `Page` e provvede:

- un metodo per eseguire una funzione di test e tornare il risultato sotto forma di html
- un metodo per contraffare ciascun tipo di dato, per esempio:
  - intero
  - stringa
  - array
  - ...
- un metodo per contraffare una richiesta http –> vedere come si può chiamare un'api php da php