boolean

CORSO FULL STACK WEB DEVELOPER

LARAVEL

Authentication

Authentication

Come funziona un Login


```
1  1  <form action="/password-check"
      method="POST">
   2
   3     <label>Username</label>
   4     <input type="text" name="user">
   5
   6     <label>Password</label>
   7     <input type="password" name="pwd">
   8
   9     <input type="submit" value="INVIA">
  10  </form>
```

Authentication

Come funziona un Login

/login

Browser utente

/password-check

Il controller riceve i dati dalla view (username e password), li controlla e decide se sono corretti

Server

Authentication

Come funziona un Login

Le informazioni contenute nel form vengono inviate in POST alla pagina /password-check.

È sicuro inviare una password in rete così come è stata inserita dall'utente? **Ovviamente no!**

Per questo si utilizzano le funzioni di hashing.
Le funzioni di hashing sono delle funzioni non invertibili che trasformano una stringa di lunghezza arbitraria in una stringa di lunghezza prefissata.

Hashing Functions

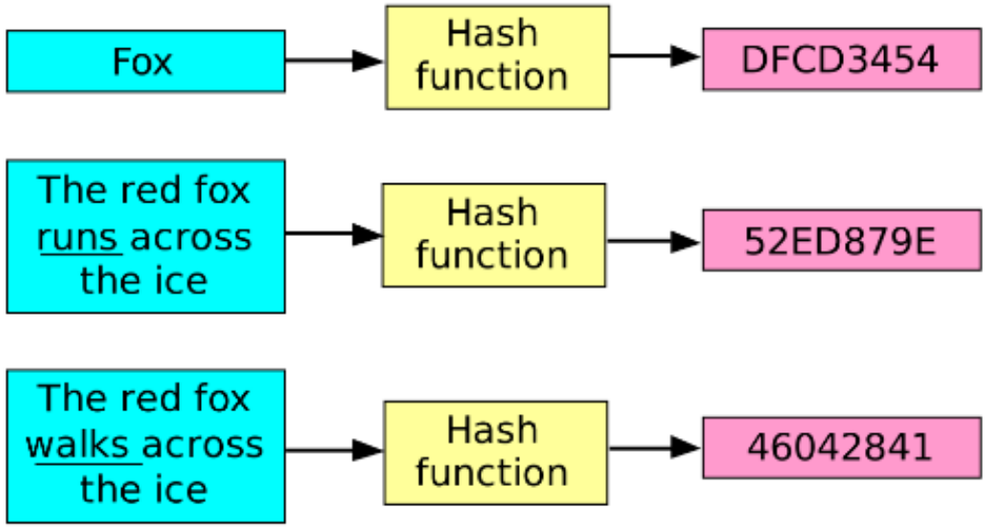
Proteggere le password

Ad ogni testo di **input** corrisponde uno e un solo output (**hash**).

Dall'**output** non è possibile ritornare al testo di input.

La funzione di hashing non è invertibile.

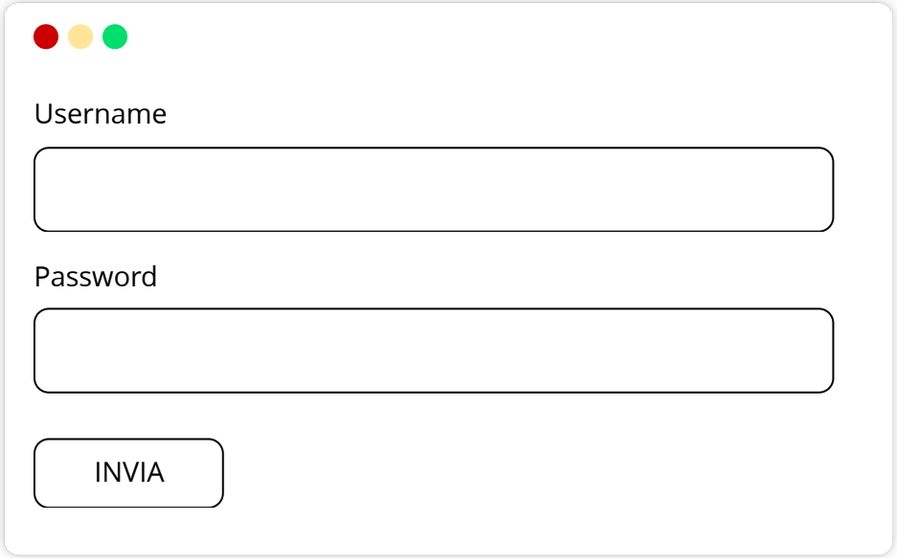
Le più utilizzate funzioni di hashing sono
SHA, MD5, bcrypt



Authentication

Login sicuro

/login



Browser utente

username=fabio

password=52EBASK2723

/password-check

Il controller riceve i dati dalla view (username e password), li controlla e decide se sono corretti

Server

Authentication

Login sicuro

Anche con JS possiamo hashare la password

```
1 1 document.getElementById('login-form').addEventListener('submit', function() {
  2     const passwordInput = document.getElementById('password');
  3     const hashedPassword = sha256(passwordInput.value);
  4     passwordInput.value = hashedPassword;
  5     return true;
  6 });
```

Authentication

E nel Backend?

Nel database non sono salvate le password “in chiaro” (o plain-text) **ma solo le versioni hashate**, rendendo molto più difficile l’utilizzo di un eventuale database rubato.

id	username	password
1	fabio	52EBASK2723
2	andrea	833JFH77SKS
3	marco	HJABS893739
4	michele	88823DAAHS8

Authentication

E nel Backend?

/password-check

Esiste un utente nella tabella users che ha username e password hashata uguali a quelli inseriti?

Se sì, l'utente è autorizzato.

tabella users

id	username	password
1	fabio	52EBASK2723
2	andrea	833JFH77SKS
3	marco	HJABS893739
4	michele	88823DAAHS8

Rainbow Tables!

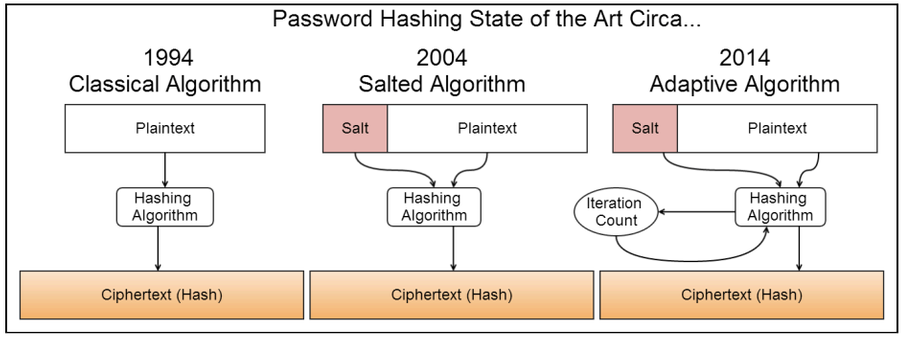
https://www.ionos.it/digitalguide/server/sicurezza/rainbow-tables/

Password salate

Usando password comuni, c'è la possibilità che un hacker possa arrivare a conoscere la password plain-text, facendo il tragitto contrario: partendo da una lista di password comuni hashate cercherà una corrispendenza con l'hash contenuto nel database.

Per evitare questo si utilizzano i **salt**.

La password memorizzata nel DB sarà:
hash(password_plaintext + salt)
rendendo impossibile la ricerca a corrispondenza inversa.



id	username	password	salt
1	fabio	52EBASK2723	KASD88SDF9
2	andrea	833JFH77SKS	ASK3948UFD
3	marco	HJABS893739	989SDFHSD9
4	michele	88823DAAHS8	092828XNSUI

Se visitate un sito e durante il recupero della password o alla registrazione vi dice in chiaro quale è la vostra password, quel sito è **ESTREMAMENTE INSICURO**

TRENITALIA
GRUPPO FERROVIE DELLO STATO ITALIANE

Invio dati registrazione

Dear **PLAIN TEXT OFFENDER,**
Please find below your User-ID and Password - referring to the email address you have given us - for the online services you have requested.

Registration summary

Userid: [redacted]
Password: RANDOMPASSWORD

Privacy

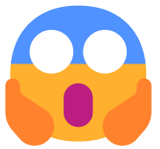
Regarding the data you have given us in order to provide you with the service, please be advised that your privacy rights are safeguarded by Trenitalia in respect of current legislation (Legislative Decree 196/2003).

TRENITALIA
GRUPPO FERROVIE DELLO STATO ITALIANE **Invio dati registrazione**

Dear **PLAIN TEXT OFFENDER,**
Please find below your User-ID and Password - referring to the email address you have given us - for the online services you have requested.

Registration summary

Userid: [redacted]
Password: RANDOMPASSWORD



Privacy

Regarding the data you have given us in order to provide you with the service, please be advised that your privacy rights are safeguarded by Trenitalia in respect of current legislation (Legislative Decree 196/2003).

E una volta autenticato?

Una volta autenticato, come fa il server a ricordarsi che l'utente si è autenticato correttamente?
Con due strumenti, **cookie** e **sessioni**:

- Il cookie è un pezzo di informazione che rimane sul browser dell'utente
- La sessione è una informazione che rimane salvata sul server

Se c'è un cookie che contiene un identificativo che corrisponde ad una sessione salvata sul server, allora l'utente è autenticato.

Laravel Magic

Con Laravel, tutta questa complessità è già gestita e pronta da utilizzare e ciò è un bene: **l'implementazione di codice che gestisce autenticazione e più in generale, sicurezza, è estremamente critico,** basta una piccola distrazione per creare bug e consentire accessi indesiderati.

Sfruttare il **codice scritto da esperti**, e visto e rivisto da vari occhi ci dà una grande sicurezza sul fatto che il codice sia **sicuro, testato ad affidabile**.

Potreste dire lo stesso del vostro codice?

Laravel Magic

Laravel gestisce/fornisce:

- Una pagina di login
- Una tabella utenti
- Tutti i controller necessari per fare login, logout, reset password
- Tutto il sistema di gestione delle sessioni per verificare se un utente è autenticato, sia nel controller che direttamente in Blade

https://laravel.com/docs/9.x/authentication

Laravel Authentication

Set up

1 Installiamo **laravel/breeze**

```
1 1 composer require laravel/breeze --dev
```

2 Creiamo lo **scaffolding di default con blade**

```
1 1 php artisan breeze:install
```

3 Modifichiamo lo **scaffolding di default per usare bootstrap**

```
1 1 # Installa preset laravel 9 bootstrap vite
  2 composer require pacificdev/laravel_9_preset
  3 # Esegui comando preset
  4 php artisan preset:ui bootstrap --auth
  5 npm i
  6 npm run dev
```

Il pacchetto che abbiamo appena istallato effettuerá una serie di modifiche al preset originale di Breeze, consentendoci di usare Bootstrap 5.x

Laravel Authentication

Dashboard Route

Laravel crea in automatico una rotta per gestire la pagina di atterraggio per gli utenti registrati dopo l'accesso (login)

Dopo la definizione della rotta, viene concatenato il metodo **middleware(['auth', 'verified']);**

Questo metodo controllerà che l'accesso sia consentito solo agli utenti loggati e verificati.

```
1  Route::get('/dashboard', function () {
2      return view('dashboard');
3  })->middleware(['auth', 'verified'])->name('dashboard');
```

Laravel Authentication

Admin Controller

Separiamo i controller dell'area di amministrazione, **creiamo un nuovo controller nel namespace Admin.**

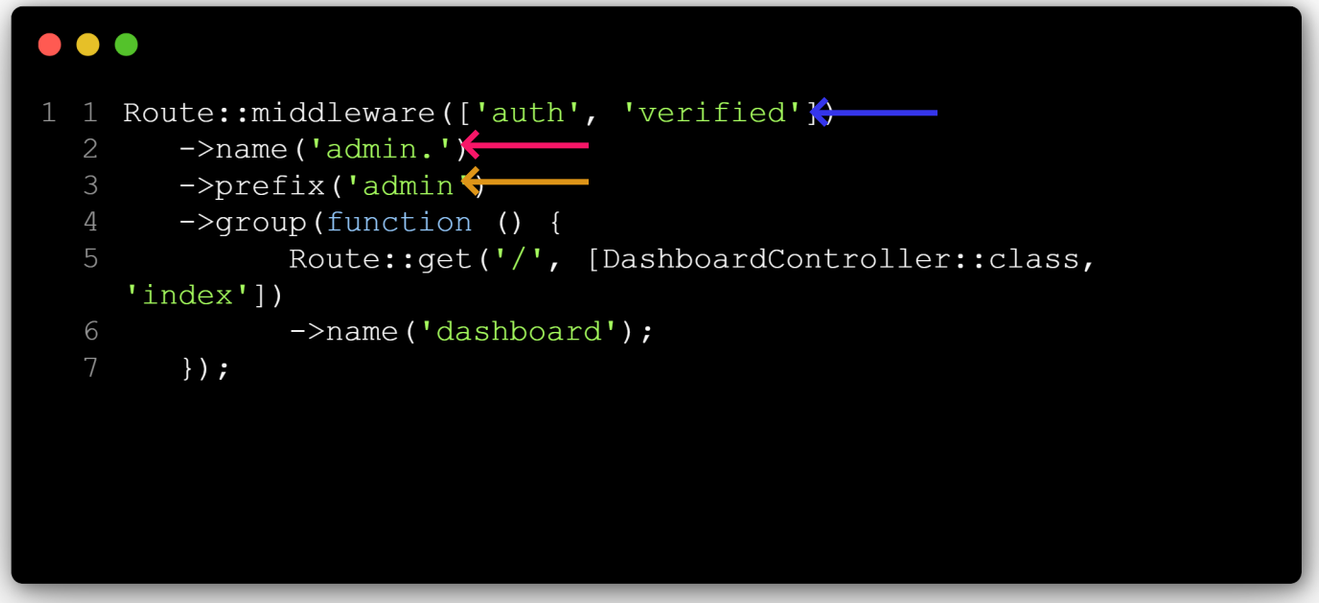
```
1 1 php artisan make:controller Admin/DashboardController
```

Laravel Authentication

Admin Routes

Raggruppiamo tutte le rotte per la parte di amministrazione del sito, in modo che abbiano:

- **middleware** 'auth e 'verified'
- **name** delle rotte che inizia con 'admin.'
- **prefix** tutti gli url iniziano con '/admin/'



```
1  Route::middleware(['auth', 'verified'])
2      ->name('admin.')
3      ->prefix('admin')
4      ->group(function () {
5          Route::get('/', [DashboardController::class,
   'index'])
6              ->name('dashboard');
7      });
```

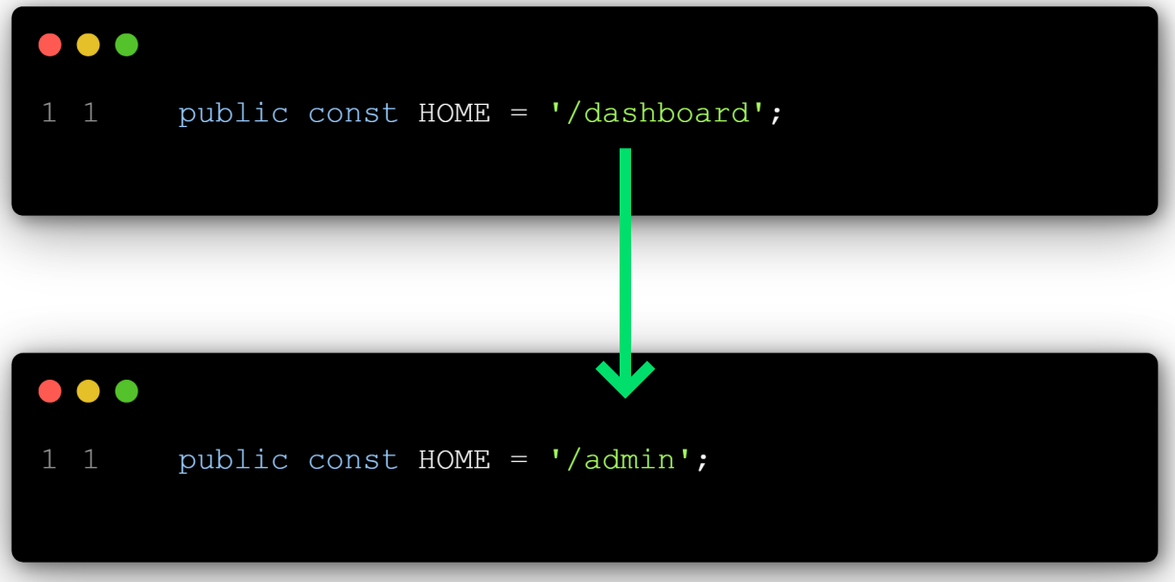
Laravel Authentication

Admin Landing Page

Dopo il login, Laravel di default reindirizza gli utenti alla rotta **/dashboard**.

Noi però vogliamo individuare l'area di amministrazione utilizzando il prefisso **/admin** per tutte le pagine di backoffice.

Possiamo modificare la path nel file **app/Providers/RouteServiceProvider.php**



Laravel Authentication

Login Redirect

Se l’utente non è autenticato, sarà dirottato automaticamente verso la pagina di login.

Questo comportamento è modificabile nel file **app/Http/Middleware/Authenticate.php**

```
1 1   protected function redirectTo($request)
  2   {
  3       if (! $request->expectsJson()) {
  4           return route('login');
  5       }
  6   }
```

Laravel Authentication

Auth User

Come facciamo a prendere i dati dell'utente loggato?

```php
use Illuminate\Support\Facades\Auth;

// Get the currently authenticated user...
$user = Auth::user();

// Get the currently authenticated user's ID...
$id = Auth::id();
```

E se vogliamo controllare se l'utente è loggato?

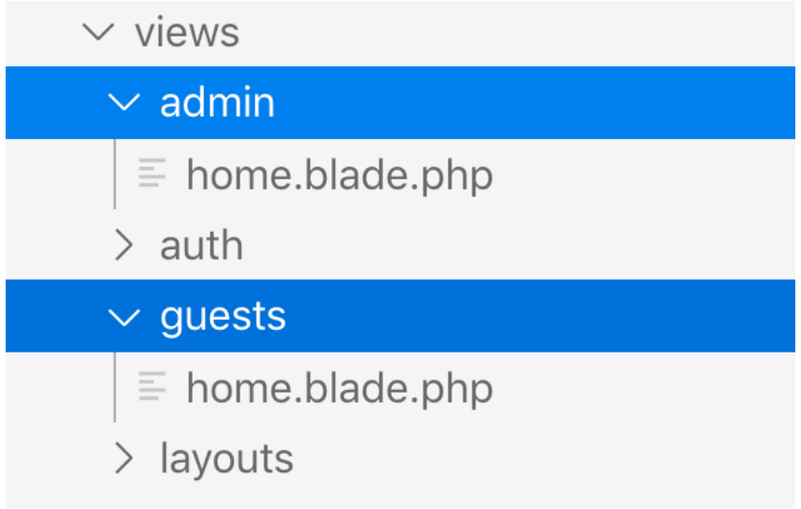
```php
use Illuminate\Support\Facades\Auth;

if (Auth::check()) {
    // The user is logged in...
}
```

Laravel Authentication

Admin vs. Guest Views

Per separare l'area di amministrazione e la parte pubblica della nostra applicazione, é possibile creare due cartelle:

- **admin** per tutte le pagine di amministrazione del **back-office**
- **guest** per le pagine pubblicamente accessibili del **front-office**



In un applicazione realizzata completamente (front/back office) con Laravel organizzeremo le views in questo modo, ma é possibile realizzare il front-office anche come un'applicazione interamente indipendente da Laravel. Lo faremo a breve.

LIVE CODING

ESERCITAZIONE