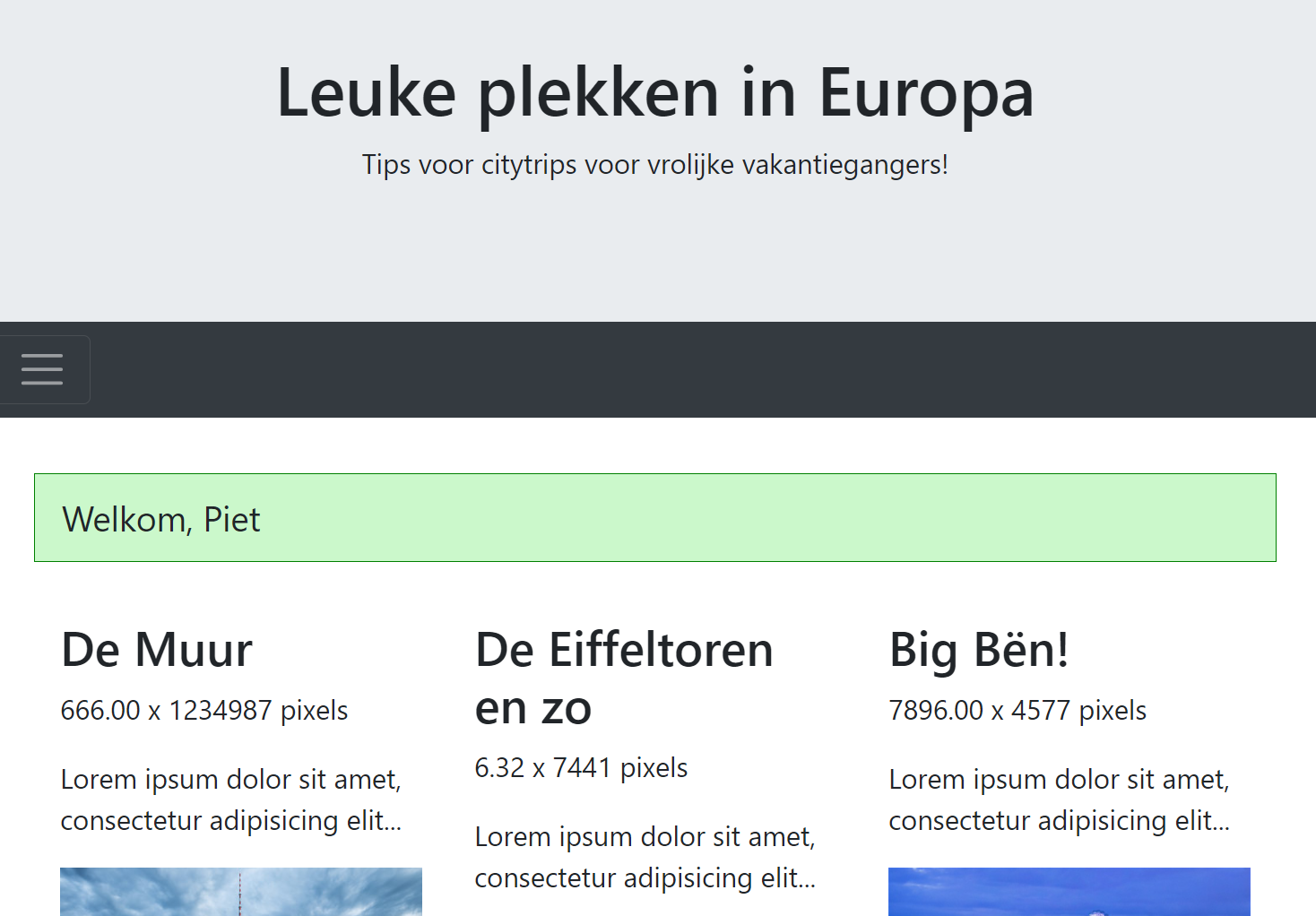
# 4.4 Access Control

Het **inloggen** in onze app werkt nog niet helemaal zoals het hoort; we willen na inloggen naar onze **homepage** gaan, of, als we niet correct inloggen, naar een iets duidelijkere en mooiere **foutpagina**.

Bovendien willen we overal controleren of de gebruiker wel ingelogd is: dat noemen we **toegangscontrole** of **access control**.

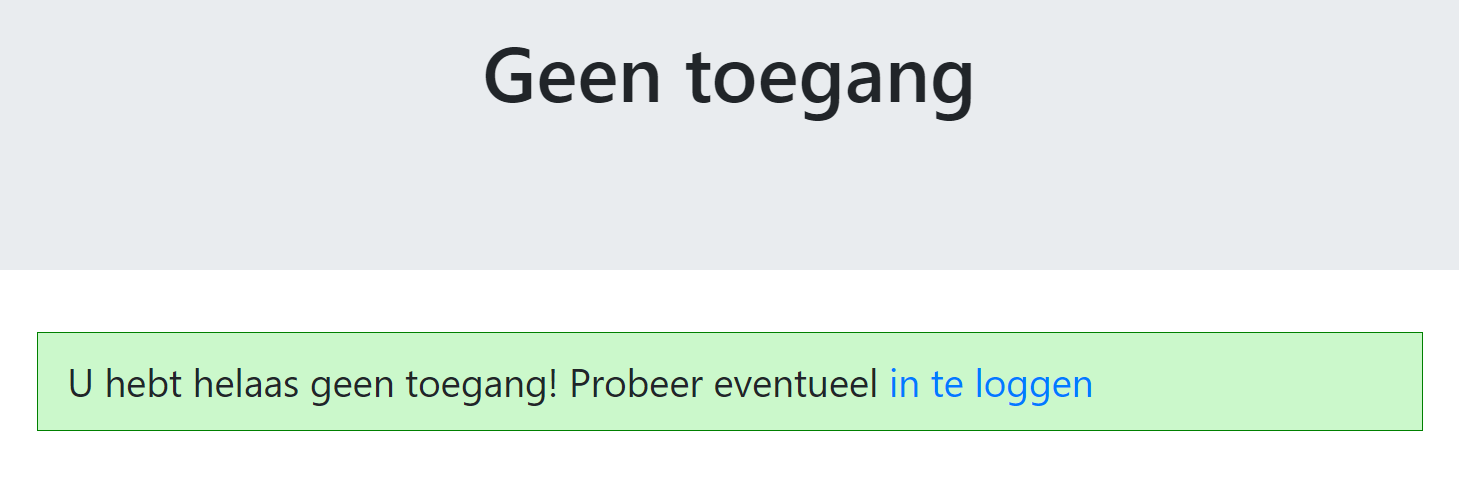
Met een geldige login, willen we naar de pagina **steden.php** gaan, en daar een welkomstbericht zien verschijnen.

**Geldige login:**



Met een **foutieve login**, gaan we naar een pagina die ons vertelt dat we **geen toegang** hebben.

**Foute login:**



Dat brengen we in orde in de volgende stappen:

**no\_access.php**

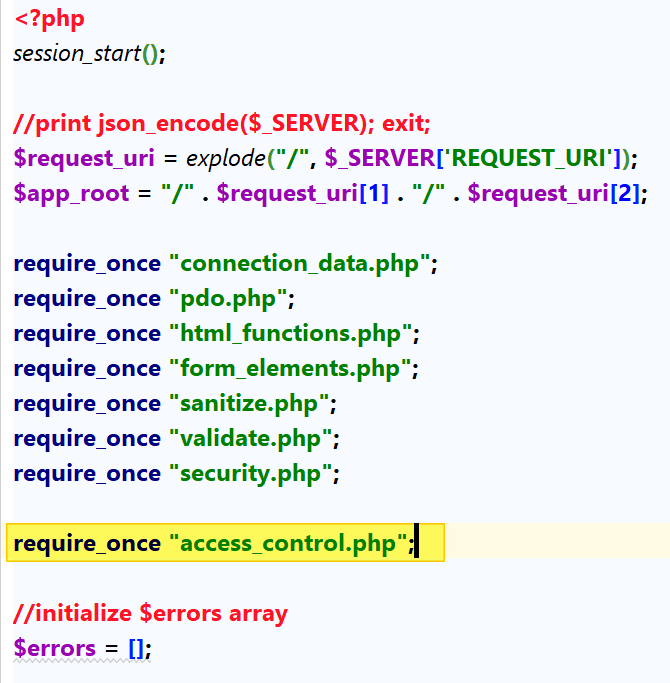
Eerst creëren we een pagina die de gebruiker zegt dat hij/zij **geen toegang** heeft (afbeelding hierboven). We slaan het script **no\_access.php** op in de hoofdmap van ons project. Het bericht is **hardcoded** (werkt dus niet met het ‘berichtensysteem’ dat we eerder invoerden), en een **template** is ook niet echt nodig: er is niets dynamisch aan deze pagina.

De link ‘**in te loggen**’ brengt de gebruiker naar **login.php**.

**lib/login\_check.php**

Dit script gaat na of de ingevoerde **e-mail** en **wachtwoord** geldig zijn. We moeten wel enkele wijzigingen aanbrengen:

* Bij een **geldige login**
  + moeten we de gebruikersgegevens (**alle velden**) opslaan in een sessievariabele **$\_SESSION['user']**. Die kunnen we dan overal controleren, ook bij volgende requests.
  + genereren we een **welkomstbericht** dat op de homepage verschijnt (deze keer wél met het berichtensysteem)
  + gaan we naar de homepage **steden.php** die dit bericht toont
* Bij een **ongeldige login**
  + maken we de sessievariabele **$\_SESSION['user']** leeg; gebruik hiervoor **unset( $\_SESSION['user'] )**
  + gaan we naar de pagina **no\_access.php**

**lib/autoload.php**

We zouden de toegangscontrole volledig in dit script kunnen schrijven, maar beter is om door te verwijzen naar een extern script **access\_control.php** dat we in de volgende stap maken.

**lib/access\_control.php:**

Dit script doet **bij elke request** de controle of de gebruiker toegang heeft tot de gevraagde pagina.

Hoe controleren we of een gebruiker ingelogd is? Door te controleren of **$\_SESSION['user']** ingevuld is. Als dat niet het geval is, redirecten we naar **no\_access.php**.

**register.php – login.php – no\_access.php**

Deze scripts moeten natuurlijk toegankelijk zijn **zonder dat je eerst moet inloggen**.

* **om te registreren** (**register.php**) moet je niet ingelogd zijn
* **om in te loggen** (**login.php**) moet je ook niet ingelogd zijn
* **om ‘geen toegang’ te krijgen** (**no\_access.php**) moet je ook niet ingelogd zijn

Nochtans krijg je momenteel geen toegang meer tot deze scripts, als je niet eerst ingelogd bent. Ze laden immers alle drie **autoload.php** en dus ook **access\_control.php**. Om in te kunnen loggen moet je dus eerst ingelogd zijn. Dat is absurd.

Hoe lossen we dat op? Maak gebruik van een globale variabele **$public\_access**. Stel die in de 3 bovenvermelde scripts met publieke toegang gelijk aan **true** voor de call naar **access\_control**.

In **access\_control.php** zelf ga je na of deze variabele **true** is. Pas wanneer dat **niet** het geval is, controleer je **$\_SESSION[‘user’]**. Geen publieke toegang en geen ingevulde gebruiker? Dan gaan we naar **no\_access.php**.

## Conclusie

In alle pagina’s/scripts die bovenaan **require\_once “autoload.php”** hebben staan, is de toegangscontrole nu ingebouwd. **Alleen ingelogde gebruikers** kunnen die pagina’s opvragen en bekijken. Tenzij we ze **expliciet publiek toegankelijk** maken met **$public\_access = true**.

Opgelet: **sessievariabelen** (zoals de **gebruikersgegevens**) blijven in Chrome geladen tot **alle vensters** van Chrome volledig gesloten worden. Een alternatief is een nieuw **Chrome Incognito venster** openen (**Ctrl-Shift-N**).

**Oplossing**: zie **oef4.4**