In our application we have a service (**ProfileService**) to manage our profile user. The service keeps the model and provides 2 methods to fetch and update data (**getProfileUser / setName**).
These methods are asynchronous, since there is a webapi request involved (we are mocking this here with a random timeout). These can sometimes return an error.

We have a component that injects the profile service and allows the user to manage the profile.

profile.service

```
import { Injectable } from '@angular/core';

export interface IProfile {
 firstName : string;
 lastName  : string;
 username   : string;
 age        : number;
}

@Injectable({ providedIn: 'root' })
export class ProfileService {
 public user: IProfile;
 constructor() {}

 getProfileUser(): Promise<IProfile> {
   return new Promise((resolve, reject) => {
     setTimeout(() => {
       if (Math.round(Math.random())) {
         this.user = {
           firstName : 'Michael',
           lastName  : 'Collins',
           username  : 'michael.collins',
           age       : 30
         };
         resolve(this.user);
       } else {
         reject({ error: 'Profile not found' });
       }
     }, Math.random() * 5000);
   });
 }

 setName(firstName: string) {
   return new Promise((resolve, reject) => {
     setTimeout(() => {
       if (Math.round(Math.random())) {
         this.user.firstName = firstName;
         resolve(this.user);
       } else {
         reject({ error: 'Invalid name' });
       }
     }, Math.random() * 5000);
   });
 }
}
```

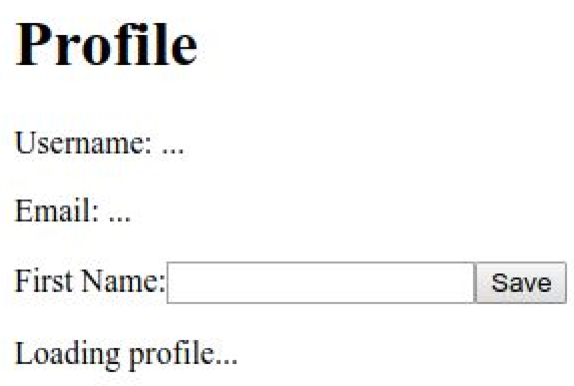
profile-settings.component.html

```
<div>
 <h1>{{title}}</h1>
 <div>
   <p>Username: ...</p>
 </div>
 <div>
   <label>First Name:</label>
   <input type="text"/>
   <input type="button" value="Save"/>
 </div>

 <p>Loading profile...</p>
 <p>Saving profile...</p>
 <p>Error!</p>

</div>
```

profile-settings.component.ts

```
import { Component, OnInit } from '@angular/core';
import {IProfile, ProfileService} from '../profile-service/profile.service';

@Component({
 selector: 'app-profile-settings',
 templateUrl: './profile-settings.component.html'
})
export class ProfileSettingsComponent implements OnInit {
 public title = 'Profile';
 public user: IProfile;

 constructor(private profile: ProfileService) { }

 ngOnInit() {}

 saveProfile() {}

}
```

The idea is to have a simple template that loads the profile object, displays the current values, and allows to change and save the name:



Profile

Username: ...

Email: ...

First Name:

Loading profile...

To do:

1. Load the profile object once the ProfileSettings component is initialized, display the username, and populate the first name into the input field. If the request results in an error, we should automatically fetch it again until we get it.
2. While the data is not ready, we should display the label **"Loading profile..."** and disable the inputs.
3. Once the data is ready, clicking the Save button we should make use of the **setName()** method and update the first name of the user in the profile. While this is happening we should display the **"Saving Profile…"** label. If the service returns an error, we should display the text **"Error!"** and the error message returned by the service. This text should be removed as soon as the value in the input changes or the "Save" button is hit again.
4. Add a new input field to update the "last name" too. There should be only 1 Save button to update both fields at the same time. This data should be sent to the profile service using the same setName() method.
5. Add a new field to the profile object: "**email**". This should be included in both the service and component. We want to display this new field in the component, but do update it.
6. In the service we need a new method **setUserEmail**(). This will do the same as **setName**(), but updating the email. The new value though will not come as a parameter, but from the current user object, so "user.**email"** = "[user.**firstName**].[user.**lastName**]@blueface.com". To make sure that email is valid we should remove any spaces from the first/last name fields.
   This method will be called automatically after **setName**() has successfully saved the first/last name, and will call the webapi the same way (use a timeout to emulate it as the others). This asynchronous process can return and error too, so the component will need to handle a new type of error coming back, displaying a different error message "Error on email generation".

As a plus:

We want to internationalize the application, so the texts cannot be hardcoded in the template. We want the title to be displayed in 3 different languages, and the option to change the current language from inside the application.