Async Validators

- Calls remote url to get result of a validation.
- A function that takes the form control value and returns an observable.

```
interface BackendResponse {
  valid: boolean;
}
export function asynchronousValidator(httpClient: HttpClient) {
  return (ctrl: AbstractControl): Observable<ValidationErrors | null> => {
    return httpClient
      .get<BackendResponse>(`remote-url/${ctrl.value}`)
      .pipe(
        map((response: BackendResponse) =>
          response.valid ? null : { asyncError: true }
        )
      );
  };
}
```

```
this.expenseForm = new FormGroup(
  {
    name: new FormControl('', {
      updateOn: 'blur',
      validators: [Validators.required, customValidator],
    }),
    date: new FormControl(''),
    amount: new FormControl(''),
    note: new FormControl('', {
      updateOn: 'blur',
      asyncValidators: [asynchronousValidator(this.httpClient)],
    }),
  },
  { validators: crossFieldValidator }
);
```

- IMPORTANT: set updateOn: ‘blur’ when validating to save BE calls.

Reactive Forms Task #5

- Financial regulators are coming after us.There are new rules that mean that some words and phrases in notes are not allowed.
- POST/**https://billing.akademija.workers.dev/** . When called with `{“toValidate”: string} returns a “{“valid”: boolean}” object, which returns if the note is valid.
- Change the expense note input field to be updated/validated on the ‘blur’ event to save backend calls.
- Create an async validator that calls the endpoint (via expensesService) and returns a validation error if it is an invalid note. (“zole” sounds like something we don’t want to buy, right?).