Diario di lavoro

Titolo progetto	Gestione ore
Luogo	Trevano-Canobbio
Data	8 ottobre 2019

Lavori svolti

Oggi la giornata è stata impiegata nella creazione del codice JavaScript che convalida i dati inseriti all'interno dei vari form creati nelle lezioni precedenti.
Questo viene fatto con l'ausilio del plugin di jQuery jQuery.validation (https://jqueryvalidation.org).

Infatti, oggi ho cominciato con il form di assegnazione di una risorsa a un lavoro: potrebbe sembrare semplice, visto che questo formulario è composto solamente da due campi, di cui uno non può essere modificato, ma questa pagina è stata una delle più complicate finora. Ho infatti dovuto controllare che il valore del campo lavoro sia rimasto quello che è stato caricato e che non sia stato modificato dall'utente, inoltre la risorsa che si intende passare al back-end deve essere un valore contenuto all'interno dei campi tra i quali è possibile scegliere, anch'essi però devono essere presi subito dopo che la pagina è stata caricata, altrimenti diventa facile per un malintenzionato modificarli e aggirare la validazione. Vista la specificità dei "requisiti", ho dovuto scrivere tutto il codice a mano e impostare che venga eseguito dopo il submit del form.

Ho poi notato che, per la seguente pagina di registrazione delle ore di lavoro, avrei dovuto validare nuovamente i campi 'lavoro' e 'risorsa' come nella pagina di assegnazione appena finita, quindi ho deciso di creare una classe JavaScript che contiene il codice per validare una o più select in modo da poterla usare per entrambe le pagine senza dover ripetere codice. Essendo una classe generica mi ci è voluto un po' per implementarla perché ho dovuto lavorare con array bidimensionali con pochi riferimenti a dei dati reali, ma alla fine sono riuscito a realizzare la classe 'SelectValidation' in allegato.

Questa classe ha reso l'implementazione di una prima parte del codice per la validazione del form nella pagina di registrazione delle ore di lavoro molto più facile e veloce rispetto alla prima "versione" della pagina di assegnazione, scritta da zero e totalmente a mano, visto che mi è bastato seguire il modello della pagina di assegnazione di una risorsa a un lavoro completata poco prima e utilizzare la nuova classe generica per la validazione dei select.

Per quanto riguarda quello che è rimasto nel file specifico alla pagina, sia per quella di assegnazione che per quella di registrazione, quello seguente è il risultato finale:

```
$(document).ready(() => {
    let selectIDs = ["#lavoro", "#risorsa"];
    let validator = new SelectValidation(selectIDs);
    $("form").on("submit", () => {
        let isLavoroValid = validator.isValueSameAsOriginal(selectIDs[0]);
        let isRisorsaValid = validator.isValueInOptions(selectIDs[1]) &&
            $("#risorsa").val().trim().length > 0;
        let lavoroErrorLabel = $("#lavoroSelect label.error");
        if (isLavoroValid) {
            //Se è stato trovato un messaggio d'errore, lo si elimina perché il valore
del lavoro è valido
            if (lavoroErrorLabel.length > 0) {
                lavoroErrorLabel.remove();
            }
            //Se il valore non è valido, aggiungi un messaggio di errore
        } else {
            //Se non viene trovato un messaggio di errore, aggiungilo
            if (lavoroErrorLabel.length === 0) {
                let lavoroErrorMessage = "Selezionare una lavoro valido";
                let errorLabel = $("<label for='lavoro' class='error'>" + lavoroErrorMe
ssage + "</label>");
                $("#lavoroSelect").find(".error-container").append(errorLabel);
            }
        }
        let risorsaErrorLabel = $("#risorsaSelect label.error");
        //Indipendentemente dalla validità o meno del valore inserito all'interno del c
ampo Lavoro
se quello inserito nel campo Risorsa è valido, rimuovi gli eventuali messaggi di errore
.
        if (isRisorsaValid) {
            //Se viene trovato un messaggio di errore, rimuovilo
            if (risorsaErrorLabel.length > 0) {
                risorsaErrorLabel.remove();
            }
            //Se invece non dovesse essere valido
        } else {
            //Cerca se è presente un messaggio di errore. In caso negativo, aggiungilo
            if (risorsaErrorLabel.length === 0) {
                let risorsaErrorMessage = "Selezionare una risorsa valida";
                let errorLabel = $("<label for='risorsa' class='error'>" + risorsaError
Message + "</label>");
                $("#risorsaSelect").find(".error-container").append(errorLabel);
            }
        }
        return isLavoroValid && isRisorsaValid;
    });
});
```

A questo punto la pagina di assegnazione era pronta, ma per finire la pagina di registrazione delle ore di lavoro, ho proseguito con la validazione dei campi della data e del numero di ore di lavoro, che è stata facile perché ho potuto usare jQuery.validate. Il risultato, infatti, è il seguente:

```
form.validate({
    rules: {
        data: {
            required: true,
            max: todayDate.toISOString()
        },
        numeroOre: {
            required: true,
            min:
                1
        }
    },
    messages: {
        data: {
            required: "Inserire una data valida",
            max:
                "La data di lavoro non può essere maggiore di quella di oggi. " +
                "Inserire un valore minore o uguale a " + todayDate.toLocaleDateString(
)
        },
        numeroOre: {
            required: "Inserire un numero di ore valido",
            min:
                "Inserire un numero di ore maggiore di 0"
        }
    }
});
```

Dove la sezione ‘rules’ contiene le regole che il plugin deve seguire per validare il form e la sezione ‘messages’ i messaggi che vanno mostrati in caso di mancata validazione.

La realizzazione di queste due pagine segna la fine della parte di implementazione delle views.

Problemi riscontrati e soluzioni adottate

Ho riscontrato un problema con la validazione dei campi di tipo ‘date’. Questo era dovuto al fatto che ho dovuto validare il campo ‘Data’ del form di registrazione delle ore di lavoro in modo che accettasse solo valori minori o uguali alla data di oggi, quindi inizialmente ho usato la seguente sintassi:

```
max: new Date()
```

Ma questo non funzionava. Ho in seguito capito che era necessario passare una stringa come valore massimo invece di un oggetto di tipo Date, come su HTML, quindi ho cambiato nel codice di seguito, che è quello definitivo:

```
max: todayDate.toISOString()
```

Dove ‘`todayDate`’ è una costante definita al di fuori della funzione che contiene la data in cui è stata caricata la pagina.

Punto della situazione rispetto alla pianificazione
Sono in anticipo rispetto alla pianificazione, visto che ho terminato in anticipo l'implementazione delle views.

Programma di massima per la prossima giornata di lavoro
Portare avanti la documentazione della pianificazione e cominciare a realizzare le classi Model che si interfacciano al database.