Diario di lavoro

Titolo progetto	Gestione ore
Luogo	Trevano-Canobbio
Data	4 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).
Piu di preciso mi sono occupato dapprima della pagina di registrazione di una risorsa, che comprende i campi "nome", una stringa di testo normale, "costo", un numero decimale con una precisione massima di 2 cifre dopo la virgola, "password", una stringa di testo normale, e infine "ruolo", una stringa di testo normale predefinita, infatti l'utente può scegliere unicamente i valori "amministratore" e "utente". Due di questi campi sono obbligatori per la registrazione, ossia "nome" e "costo".
Il codice generato è il seguente (per ulteriori dettagli vedi la sezione "Problemi riscontrati e soluzioni adottate"):

```
$(document).ready(() => {
    let form = $("form");
    form.validate({
        rules: {
            nome: {
                required: true,
                maxlength: {
                    param: 255
                }
            },
            costo: {
                required: true,
                min: {
                    depends: (element) => {
                        return (element.value * 1) < 0;
                    }
                }
            },
            password: {
                maxlength: {
                    param: 255
                }
            }
        },
        messages: {
            nome: {
                required: "Inserire un nome valido",
                maxlength: "Inserire un nome più corto di 256 caratteri"
            },
            costo: {
```

```
                required: "Inserire un costo",
                min: "Inserire un costo maggiore o uguale a 0"
            },
            password: {
                maxlength: "Inserire una password più corta di 256 caratteri"
            }
        }
    });
    …
});
```

Quelle mostrate sopra sono solo le regole di base per la validazione lato client, cose come “questo campo è obbligatorio da riempire” e “lunghezza massima consentita per il valore inserito = 255 caratteri”. Il limite di 255 caratteri per tutti i campi che contengono testo deriva dalla progettazione del database, nella quale ho inserito che, visto che il tipo varchar di MySQL alloca la memoria dinamicamente in base al numero di caratteri da memorizzare, il massimo di caratteri è un numero grande come 255 perché in ogni caso non vi sono sprechi di memoria.
La sezione “messages” contiene i messaggi di errore da mostrare all’utente in caso di mancata validazione dei dati inseriti: a ogni campo impostato sopra è assegnata una stringa di testo da far vedere.

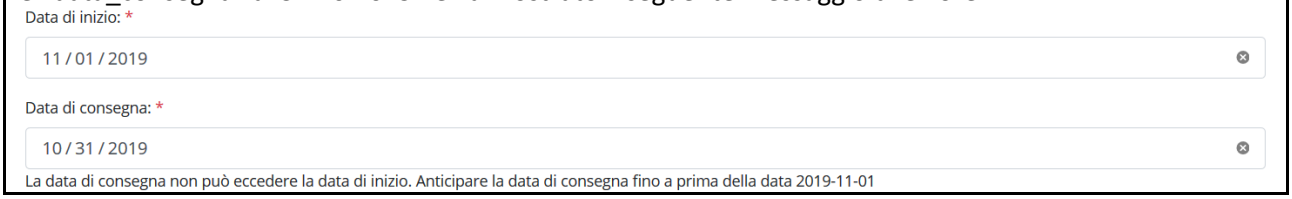
Dopodiché ho realizzato la validazione del form di inserimento di un nuovo lavoro, che comprende i campi “nome”, una stringa di testo normale, “ore”, un numero intero positivo, “data_inizio”, una data normale, “data_fine”, una data che non può essere minore di quella inserita all’interno di “data_inizio”, e “note”, una stringa di testo normale.
Il risultato è il seguente codice:

```
$(document).ready(() => {
    $("form").validate({
        rules: {
            nome: {
                required: true,
                maxlength: {
                    param: 255
                }
            },
            data_inizio: {
                required: true
            },
            data_consegna: {
                required: true,
                min: () => {
                    return $("#data_inizio").val();
                }
            },
            ore: {
                required: true,
                min: {
                    param: 1
                }
            },
```

```
            note: {
                maxlength: {
                    param: 255
                }
            }
        },
        messages: {
            nome: {
                required: "Inserire un nome valido",
                maxlength: "Inserire un nome più corto di 256 caratteri"
            },
            ore: {
                required: "Inserire un numero di ore preventivate valido",
                min: "Inserire un numero di ore preventivate maggiore di 0"
            },
            data_inizio: {
                required: "Inserire una data di inizio"
            },
            data_consegna: {
                required: "Inserire una data di consegna",
                min: "La data di consegna non può eccedere la data di inizio. Anticipar
e la data di consegna fino a prima della data {0}"
            },
            note: {
                maxlength: "Inserire delle note per un totale di meno di 256 caratteri"
            }
        }
    });

});
```

Di interessante in questo codice c'è solo la regola per il valore minimo consentito impostata al campo "data_consegna", che viene presa dinamicamente dal valore inserito nel il campo "data_inizio". È anche importante notare la notazione "`{0}`" inserita all'interno del messaggio di errore per lo stesso campo, infatti "`{0}`" è un segnaposto che permette al messaggio di essere riempito dinamicamente con il valore impostato nello stesso campo. In parole povere questo significa che se imposto "data_inizio" al 1.11.2019 e "data_consegna" al 31.10.2019 verrà mostrato il seguente messaggio di errore:



Problemi riscontrati e soluzioni adottate

Ho avuto un problema con l'implementazione del controllo sul valore per il costo, nello specifico per la verifica che il valore fosse maggiore di 0: non essendo familiare con il plugin ci ho messo un attimo per capire come andasse fatto un accertamento del genere. Alla fine ho usato le opzioni personalizzate nel modo seguente:

```
form.validate({
    rules: {
        …
        costo: {
            required: true,
            min: {
                depends: (element) => {
                    return (element.value * 1) < 0;
                }
            }
        }
       …
     }
}
```

Per lo stesso motivo ho fatto fatica ad implementare il controllo per verificare che il ruolo sia specificato se lo è pure la password. Stavolta ho utilizzato una funzione apposta in questo modo:

```
//Variabile che dice se il ruolo è stato validato con esito negativo (quindi contiene g
ià un messaggio di errore)
var isRuoloValid = true;
form.on("submit", (evt) => {
    //Ottieni il valore del radio button selezionato:
    //Prendi tutti i radio button del ruolo
    let ruoloRadios = $("input[name='ruolo']");
    //Istanzia la variabile che conterrà il valore
    let ruoloInput = "";
    for (let i = 0; i < ruoloRadios.length; ++i) {
        let ruoloRadio = ruoloRadios[i];
        //Cicla tutti i radio button e trova quello selezionato
        if (ruoloRadio.checked) {
            ruoloInput = ruoloRadio.value.trim();
        }
    }
```

```
    let radioDiv = $("#ruolo");
    //Controlla se la password è stata assegnata
    if ($("#password").val().trim().length > 0) {
        //Se è stato selezionato un valore per il ruolo
        if (ruoloInput.length > 0) {
            //Se c'è un messaggio di errore perché il ruolo non è valido
            if (!isRuoloValid) {
                //Rimuovilo
                radioDiv.find("label.error").remove();
            }
            isRuoloValid = true;
        } else {
            //Non è stato assegnato un valore!
            //Aggiungi un messaggio di errore
            let ruoloMsg = "Specificare un ruolo per l'utente";
            let errorLabel = $("<label for=\"ruolo\" class=\"error\">" + ruoloMsg + "</
label>");
            radioDiv.append(errorLabel);
            //NON SUBMITTARE IL FORM
            evt.preventDefault();
            isRuoloValid = false;
        }
    } else {
        //La password non è stata specificata: nessun controllo
        isRuoloValid = true;
    }
    //Concretamente, ritorna falso solo se il form non è valido, questo evita che il fo
rm submitti e cambi pagina
    return isRuoloValid;
});
```

Punto della situazione rispetto alla pianificazione
Sono in anticipo visto che ho deciso di installare XAMPP al posto dei componenti singoli per il web server. Inoltre mi ci è voluto molto meno di quanto programmato per l'installazione e la configurazione di PHPStorm, Xdebug e del modulo su PHPStorm del progetto.

Programma di massima per la prossima giornata di lavoro
Portare avanti la documentazione della pianificazione e la stesura del codice JavaScript che verifica la validità dei dati inseriti all'interno dei vari form presenti sulla piattaforma.