APPLICAZIONI WEB: LINGUAGGI E ARCHITETTURE
UNIVERSITA' DEL PIEMONTE ORIENTALE – AMEDEO AVOGADRO
A.A. 2020-2021

ISTRUZIONI PER IL SETUP E SCELTE IMPLEMENTATIVE

Per eseguire il progetto bisogna avere un IDE che consenta l'esecuzione del progetto come Visual Studio 2019. Una volta avviato l'IDE si avvia sia lo Slave che il Master è possibile lanciare un comando sul Master cliccando su Run all'altezza di uno dei job già preimpostati.

Le specifiche richiedevano di implementare un job scheduler in grado di lanciare processi (i job) in modo temporizzato su uno o più nodi.

Il progetto consiste di due applicazioni web: un master e uno slave. Il master consente all'utente di poter scegliere i comandi da far eseguire allo slave. Nell'applicazione Master è possibile registrarsi come nuovo utente con la possibilità di fare log in. Un nuovo utente può anche esssere creato una volta acceduto come Admin. Il processo di lancio di un job può essere posticipato tramite l'utilizzo di Cron. Sono previste due tipologie di utenti: Admin e Editor. L'applicazione Master ha anche altre funzionalità: consentire la visualizzazione dei logs da parte degli utenti, la modificazione dei ruoli degli utenti da parte dell'Admin. Per creare un nuovo job, nodo, gruppo o associazione tra gruppi e nodi bisogna premere nella relativa view il pulsante verde in alto a sinistra. È possibile modificare o cancellare job, nodi e gruppi cliccando su edit o delete. Gli utenti possono essere modificati solo dagli utenti admin.

Ho iniziato a sviluppare il progetto implementando lo slave, una API che consente di eseguire un qualsiasi comando.

Il Master, come lo Slave, è stato sviluppato partendo dalle videolezioni e dagli esempi di applicazioni resi disponibili dal docente. Per la parte di progetto inerente al Master, mi sono occupato di creare il DB e di rappresentarlo nel programma con il modello con l'aiuto di EntityFramework. Partendo dalle classi modello ho proceduto nello sviluppo delle viste e dei controller. Una volta creati i controller li ho modificati per ottenere dal programma i comportamenti definiti dalle specifiche. Infine, ho

proceduto modificando leggermente le view al fine di avere un risultato più idoneo alle esigenze degli utenti.

Il progetto è diviso secondo il pattern MVC (Model View Controller):

- Il modello è implementato nel package Models che contiene le classi: AspNetRoleClaims, AspNetRoles, AspNetUserClaims, AspNetUserLogins, AspNetUserRoles, AspNetUsers, AspNetUserTokens, GroupsNodes, Nodes, Jobs, Logs, Roles, Users. Queste classi come da definizione di modello rappresentano lo stato delle tabelle nel DB e sono utilizzate per interagire con esso.
- Le viste sono implementate nel package Views in cui ci sono cartelle per ogni vista utilizzata nel progetto: Account, GroupsNodes, Nodes, Jobs, Logs, Roles, Shared, Users.
- I controller che si occupano di gestire chiamate http, di interagire con il DB e di renderizzare le views sono implementati nel package Controllers.

Tra gli altri package ci sono:

- Areas inerente alla registrazione degli utenti
- Data contiene il codice riguardante la creazione dell'utente di default e la classe contesto del modello
- Migrations contiene le migrazioni per aggiornare il DB partendo dal codice, quindi con un approccio code-first.
- ViewModels che contiene le classi intermediarie tra la vista e il modello, riguardanti all'utente, ai ruoli e al login.

MODELLO DEI DATI