UNIVERSITÀ
DI PARMA

Dipartimento di Scienze Matematiche, Fisiche e Informatiche

Corso di Laurea [Triennale in Informatica]

Titolo in Italiano

Titolo in Inglese

Candidato:
Dennis Turco

Relatore:
Prof. Nome Cognome

Correlatori:
Prof. co-Nome co-Cognome
Prof. co-Nome2 co-Cognome2

Anno Accademico 2020–2023

Dedica

Indice

Elenco delle figure

Elenco degli algoritmi

Elenco delle tabelle

Introduzione

Nel seguente elaborato tratterò una relazione relativa al progetto di tirocinio svolto presso un'azienda software house "ISolutions" di Noceto.

Il progetto si concentra sulla creazione di una piattaforma di e-learning per la gestione del processo di OnBoarding aziendale. L'obiettivo è di fornire ai nuovi dipendenti un'esperienza guidata, strutturata e personalizzata attraverso la piattaforma web.

Attualmente, l'azienda gestisce tra 20 e 30 processi di OnBoarding dei nuovi dipendenti ogni anno, grazie al personale HR, utilizzando un foglio di calcolo Excel per la gestione dei dati e del completamento dei vari task. Tuttavia questo metodo si basa su un funzionamento manuale di aggiornamento dei task completati dai vari dipendenti e richiede una supervisione umana costante, il che lo rende inefficiente e non scalabile. Inoltre, le informazioni raccolte attraverso il modulo post-OnBoarding non sono sempre esaustive e dettagliate. A causa dell'espansione dell'azienda, questo sistema sta diventando sempre più inefficiente e rischioso. Per questo motivo risulta importante l'implementazione di un software dedicato al processo di OnBoarding dei dipendenti, con lo scopo di garantire una riduzione del rischio di errori e di migliorare l'esperienza generale. Inoltre, il software deve permettere di automatizzare alcune attività ripetitive, liberando tempo prezioso per il team HR, che potrà concentrarsi su attività di maggiore valore aggiunto per l'azienda. Infine, il nuovo sistema di OnBoarding ha la necessità di consentire l'acquisizione e l'archiviazione in maniera più sicura e organizzata dei dati dei dipendenti, rispettando le normative sulla privacy e semplificando le attività di audit interno.

Il processo di OnBoarding avrà una durata di circa un mese per uno sviluppatore junior e due settimane per uno sviluppatore senior. La piattaforma guiderà i nuovi dipendenti attraverso i vari task previsti in modo strutturato e personalizzato, fornendo loro gli strumenti necessari per acquisire le cono-

scenze e le competenze richieste per il loro ruolo.
Inoltre, la piattaforma fornirà un modulo per la raccolta di feedback post-OnBoarding. Il modulo sarà strutturato in modo da raccogliere informazioni dettagliate e utili per migliorare continuamente il processo di OnBoarding. Questo feedback sarà utilizzato per aggiornare e migliorare costantemente la piattaforma.
Infine, poiché l'azienda è certificata ISO 9001 (certificazione della qualità), la piattaforma fornirà un'ulteriore valutazione del processo di OnBoarding. Dopo che il dipendente ha completato il processo, verrà chiesto di esprimere un giudizio attraverso un modulo dedicato. Ciò permetterà all'azienda di comprendere se il processo di OnBoarding sta funzionando correttamente e se ci sono aree che possono essere migliorate.
In sintesi, la piattaforma di e-learning per la gestione del processo di OnBoarding aziendale rappresenta un'importante innovazione per l'azienda. Grazie alla sua efficienza e scalabilità, la piattaforma migliorerà l'esperienza del dipendente e ridurrà il carico di lavoro del personale HR. Inoltre, la raccolta di feedback dettagliati post- OnBoarding e la valutazione attraverso il modulo dedicato forniranno all'azienda le informazioni necessarie per migliorare costantemente il processo di OnBoarding.

Il processo pricincipale di OnBoarding aziendale deve prevedere le seguenti fasi.

- Introduzione (˜10 tasks);
- Setup della work station (˜20 tasks);
- Documentazione, sia amministrativa che tecnica (˜30 tasks);
- Video introduttivi, tecnici e orientati ai prodotti sviluppati (˜10 tasks);
- Overview dell'organizzazione aziendale e degli strumenti utilizzati (˜10 tasks);
- Hands On degli applicativi aziendali (˜20 tasks);
- Formazione sui principali linguaggi di programmazione utilizzati (˜20 tasks);
- Overview struttura codice delle soluzioni software (˜10 tasks);
- Test finale con revisione e valutazione.

Sono previsti inoltre alcuni step preliminari, in carico al team HR, per la predispozione di quanto necessario (creazione utenza, preparazione pc etc. . .), e una fase finale di retrospettiva per raccogliere feedback riguardo il processo dal neo assunto.

Alcune desiderate del progetto:

- Integrazione autenticazione con modulo di login e gestione livelli di autorizzazione;
- Pannello amministrativo per il team HR per gestire (creazione, modifica, eliminazione) dei vari tasks;
- Tracciamento tempo impiegato sui vari task e reportistica per team HR;
- Possibilità di avviare, sospendere e saltare un task;
- Integrazione fase finale di test, revisione e valutazione.

Il progetto di creazione della piattaforma di e-learning per la gestione del processo di OnBoarding aziendale è stato realizzato come web application .NET MVC 6.0 utilizzando molteplici tecnologie e linguaggi di programmazione tra cui: C#, HTML, Css, Javascript, SQL.

- Capitolo 1 → Le caratteristiche dell'OnBoarding.
- Capitolo 2 → Il modello dell'OnBoarding nell'azienda.
- Capitolo 3 → Le tecnologie utilizzate.
- Capitolo 4 → Il progetto ed il suo sviluppo.
- Capitolo 5 → Risultati e sviluppi futuri.

Capitolo 1

Le caratteristiche dell'OnBoarding

Capitolo 2

Il modello dell'OnBoarding nell'azienda

Capitolo 3

Le tecnologie utilizzate

Capitolo 4

Il progetto e il suo sviluppo

4.1 Database

Una parte sostanziale del nostro impegno aziendale per portare a compimento il progetto si è concentrata in modo dettagliato sulla creazione del database, ritenuto elemento cruciale per assicurare il pieno funzionamento della piattaforma.

La creazione del database è stata articolata in diverse fasi chiave:

- Progettazione della struttura del database, un processo attentamente studiato.
- Scrittura del database in linguaggio SQL, una tappa essenziale.
- Implementazione delle interrogazioni al database nella sezione di backend, facendo uso della libreria Dapper.

4.1.1 Progettazione

La fase di progettazione 4.1 del database ha occupato un ruolo fondamentale nel corso di questo processo, coinvolgendo un'analisi costante e una riflessione profonda. Il nostro obiettivo principale era plasmare un database estremamente completo, in grado di soddisfare appieno le esigenze della piattaforma di e-learning. In aggiunta, ci siamo concentrati su:

- Migliorare la leggibilità del database;
- Rendere più agevole la manutenzione;
- Fornire ampie possibilità di estensione del sistema.

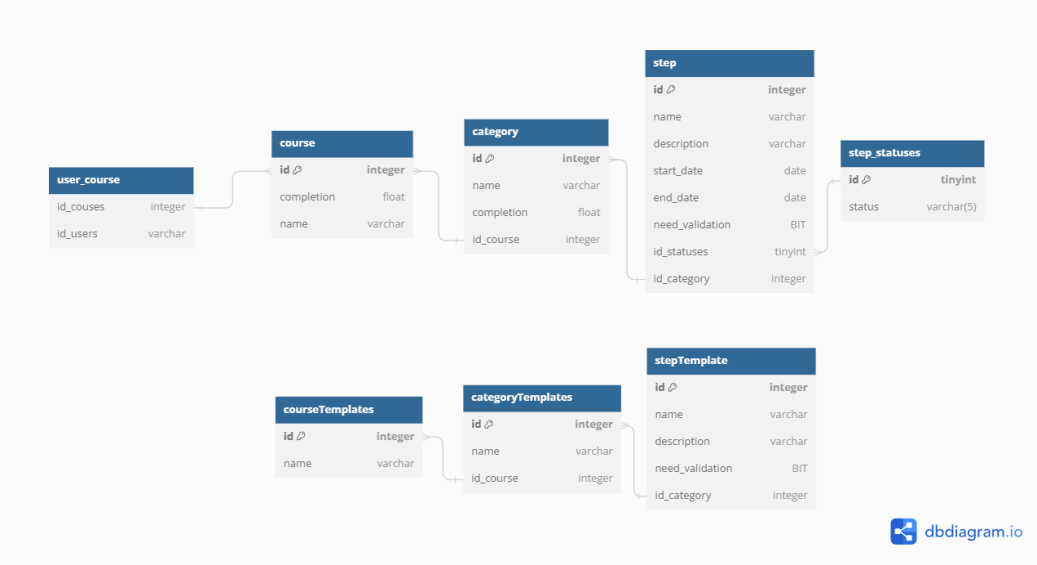

Figura 4.1: progettazione del database

Nota: la progettazione mostrata non tiene traccia delle tabelle utente, perchè vengono gestite automaticamente dal progetto .NET grazie al servizio di autenticazione già incluso (libreria `Microsoft.EntityFrameworkCore.Migrations;`).

4.1.2 Scrittura nel linguaggio SQL

Il risultato finale di questa fase è il seguente (non sono riportate le tabelle generate dal servizio di Autenticazione, perchè gestite automaticamente dalla libreria a disposizione):

```
CREATE TABLE [dbo].[Course] (
    [Id]          INT             IDENTITY (1, 1) NOT NULL,
    [Name]        NVARCHAR (255)  NOT NULL,
    [Completion] REAL             DEFAULT ((0)) NULL,
    [StartDate]  DATETIME2 (7)   NULL,
    [EndDate]    DATETIME2 (7)   NULL,
    CONSTRAINT [PK_Course] PRIMARY KEY CLUSTERED ([Id] ASC)
);

CREATE TABLE [dbo].[Category] (
    [Id]          INT             IDENTITY (1, 1) NOT NULL,
    [Name]        NVARCHAR (255)  NOT NULL,
    [Completion] REAL             DEFAULT ((0)) NULL,
    [IdCourse]    INT             NOT NULL,
    CONSTRAINT [PK_Category] PRIMARY KEY CLUSTERED ([Id] ASC),
```

```
    CONSTRAINT [FK_Category_Course_IdCourse]
  FOREIGN KEY ([IdCourse]) REFERENCES [dbo].[Course] ([Id])
  ON DELETE CASCADE
   );
   GO
   CREATE NONCLUSTERED INDEX [IX_Category_IdCourse]
     ON [dbo].[Category]([IdCourse] ASC);

-- 3 stati:
  -- 1. done
  -- 2. todo
  -- 3. check
CREATE TABLE [dbo].[StepStatus] (
  [Id]     INT            IDENTITY (1, 1) NOT NULL,
  [Status] NVARCHAR (10) NOT NULL,
  CONSTRAINT [PK_StepStatus] PRIMARY KEY CLUSTERED ([Id] ASC)
);

CREATE TABLE [dbo].[Step] (
  [Id]             INT            IDENTITY (1, 1) NOT NULL,
  [Name]           NVARCHAR (255) NOT NULL,
  [Description]    NVARCHAR (MAX) NULL,
  [StartDate]      DATETIME2 (7)  NULL,
  [EndDate]        DATETIME2 (7)  NULL,
  [Lock]           BIT            DEFAULT ((1)) NULL,
  [NeedValidation] BIT            DEFAULT ((0)) NULL,
  [IdStatus]       INT            DEFAULT ((1)) NULL,
  [IdCategory]     INT            NOT NULL,
  CONSTRAINT [PK_Step] PRIMARY KEY CLUSTERED ([Id] ASC),
  CONSTRAINT [FK_Step_StepStatus_IdStatus]
FOREIGN KEY ([IdStatus]) REFERENCES [dbo].[StepStatus] ([Id])
ON DELETE CASCADE,
  CONSTRAINT [FK_Step_Category_IdCategory]
FOREIGN KEY ([IdCategory]) REFERENCES [dbo].[Category] ([Id])
ON DELETE CASCADE
);
GO
CREATE NONCLUSTERED INDEX [IX_Step_IdCategory]
  ON [dbo].[Step]([IdCategory] ASC);
GO
CREATE NONCLUSTERED INDEX [IX_Step_IdStatus]
  ON [dbo].[Step]([IdStatus] ASC);

```

```
CREATE TABLE [dbo].[UserCourse] (
  [UserId]   NVARCHAR (450) NOT NULL,
  [CourseId] INT            NOT NULL,
  CONSTRAINT [FK_UserCourse_Course_CourseModel]
FOREIGN KEY ([CourseId]) REFERENCES [dbo].[Course] ([Id])
ON DELETE CASCADE,
  CONSTRAINT [FK_UserCourse_AspNetUsers_UserId]
FOREIGN KEY ([UserId]) REFERENCES [dbo].[AspNetUsers] ([Id])
ON DELETE CASCADE
);
GO
CREATE NONCLUSTERED INDEX [IX_UserCourse_CourseModel]
    ON [dbo].[UserCourse]([CourseId] ASC);
GO
CREATE NONCLUSTERED INDEX [IX_UserCourse_UserId]
    ON [dbo].[UserCourse]([UserId] ASC);

CREATE TABLE [dbo].[CourseTemplate] (
    [Id]             INT            IDENTITY (1, 1) NOT NULL,
    [Name]           NVARCHAR (255) NOT NULL,
    [Creator]        NVARCHAR (255) NOT NULL,
    [CreationDate]   DATETIME2 (7)  NULL,
    [LastUpdateDate] DATETIME2 (7)  NULL,
    CONSTRAINT [PK_CourseTemplate] PRIMARY KEY CLUSTERED ([Id] ASC)
);

CREATE TABLE [dbo].[CategoryTemplate] (
    [Id]                INT         IDENTITY (1, 1) NOT NULL,
    [name]              NCHAR (255) NOT NULL,
    IDCourseTemplate INT         NOT NULL,
    PRIMARY KEY CLUSTERED ([Id] ASC),
    FOREIGN KEY (IDCourseTemplate)
 REFERENCES [dbo].[CourseTemplate] ([Id])
 ON DELETE CASCADE
);

CREATE TABLE [dbo].[StepTemplate] (
    [Id]                 INT            IDENTITY (1, 1) NOT NULL,
    [Name]               NVARCHAR (255) NOT NULL,
    [Description]        NVARCHAR (MAX) NOT NULL,
    [NeedValidation]     BIT            DEFAULT ((0)) NULL,
    [IDCategoryTemplate] INT            NOT NULL,
    CONSTRAINT [PK_StepTemplate] PRIMARY KEY CLUSTERED ([Id] ASC),
```

```
102     CONSTRAINT [FK_StepTemplate_CategoryTemplate_IDCategoryTemplate
        ]
103  FOREIGN KEY ([IDCategoryTemplate]) REFERENCES [dbo].[
        CategoryTemplate] ([Id])
104  ON DELETE CASCADE
105 );
106 GO
107 CREATE NONCLUSTERED INDEX [IX_StepTemplate_IDCategoryTemplate]
108     ON [dbo].[StepTemplate]([IDCategoryTemplate] ASC);
```

Codice 4.1: traduzione della progettazione del database nel linguaggio SQL

4.1.3 Interrogazioni al Database

Inizialmente, le interrogazioni al database erano state sviluppate attraverso stored procedure. Tra i numerosi vantaggi proposti, l'idea principale era ottenere:

- Riutilizzo del codice;
- Semplificazione della manutenzione;
- Prestazioni migliorate;

Tuttavia, durante lo sviluppo dell'applicativo, grazie al suggerimento di alcuni membri dell'azienda, si è preferito sostituire le procedure con interrogazioni dirette dal codice tramite il framework Dapper, al fine di poter migliorare:

- Prestazioni in termini di tempo;
- Semplificare il debugging del codice;
- Centrallizzare l'intera logica in un unico posto.

All'interno della piattaforma si possono distinguere 3 macro categorie di interrogazioni per:

- Inserimento dei dati per aggiungere:
  - corsi;
  - categorie (sotto tipo dei corsi);
  - step (sotto tipo delle categorie);
  - corsi template;
  - categorie template (sotto tipo dei corsi template);

  - step template (sotto tipo delle categorie template).

- Eliminazione dei dati per togliere:
  - corsi;
  - categorie (sotto tipo dei corsi);
  - step (sotto tipo delle categorie);
  - corsi template;
  - categorie template (sotto tipo dei corsi template);
  - step template (sotto tipo delle categorie template).
- Lettura dei dati per permettere di ottenere tutte le informazioni necessarie per la generazione corretta della pagina dinamica dato un determinato utente connesso;

4.2 Scrittura del codice C#

Nel contesto dello sviluppo della parte back-end del progetto, è stato optato l'impiego del linguaggio di programmazione C#. Questa scelta si è rivelata fondamentale per la gestione della logica e il comportamento del sito web. L'obbiettivo principale è stato garantire una gestione efficiente del database direttamente attraverso il codice, sfruttando appieno le potenzialità del framework Dapper.
Utilizzando C# e il framework Dapper, è stato possibile realizzare un corretto collegamento tra dati e utenti. Questo ha reso possibile la generazione dinamica e accurata delle pagine web, consentendo un'esperienza utente ottimale.

4.3 Layout della piattaforma

Capitolo 5

Risultati e sviluppi futuri

Conclusione

Conclusione che riassume il lavoro svolto ed eventuali lavori futuri.

Bibliografia

[Allen and Zilberstein, 2009] Allen, M. and Zilberstein, S. (2009). Complexity of decentralized control: Special cases. In *23rd Annual Conference on Neural Information Processing Systems 2009, 7-10 December , Vancouver, British Columbia, Canada*, pages 19–27. Curran Associates, Inc.

[Bernstein et al., 2002] Bernstein, D. S., Givan, R., Immerman, N., and Zilberstein, S. (2002). The complexity of decentralized control of markov decision processes. *Mathematics of operations research*, 27(4):819–840.

[De Weerdt et al., 2003] De Weerdt, M., Bos, A., Tonino, H., and Witteveen, C. (2003). A resource logic for multi-agent plan merging. *Annals of Mathematics and Artificial Intelligence*, 37(1-2):93–130.

[Durfee, 2001] Durfee, E. H. (2001). *Distributed Problem Solving and Planning*, pages 118–149. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Gelfond and Lifschitz, 1998] Gelfond, M. and Lifschitz, V. (1998). Action languages. *Electron. Trans. Artif. Intell.*, 2:193–210.

[Russell and Norvig, 2010] Russell, S. J. and Norvig, P. (2010). *Artificial Intelligence - A Modern Approach, Third International Edition*. Pearson Education.

Ringraziamenti

Appendice A

Appendice di Esempio