13/06/2020

Legislature Repubbilca italiana

ITIA – ESAME DI STATO DI ISTRUZIONE SECONDARIA SUPERIORE (quesito 2)

Martino Papa VAIs

Descrizione Progetto - A

Il quesito affrontato riguarda la progettazione di una base di dati avente lo scopo di contenere le informazioni riguardanti legislature della Repubblica italiana, parlamentari, leggi emanate e collegi elettorali.

Una cosa importante da considerare è come tali informazioni siano tutte di dominio pubblico e che quindi non si potrà incorrere in problemi di privacy o di accesso a dati secretati, tuttavia sarà fondamentale garantire la correttezza e l'integrità delle informazioni evitando quindi modifiche o cancellazioni indesiderate.

Bisognerà inoltre prevedere l'eventualità di un'elevato numero di accessi alla base di dati, il servizio sarà infatti pubblico e utilizzabile da tutti i cittadini.

Tecnologie Utilizzate

Il linguaggio scelto per la progettazione della base di dati è stato SQL. MySQL Server si occupa dell'hosting del database la quale realizzazione è avvenuta nell'ambiente di sviluppo offerto da MySQL workbench.

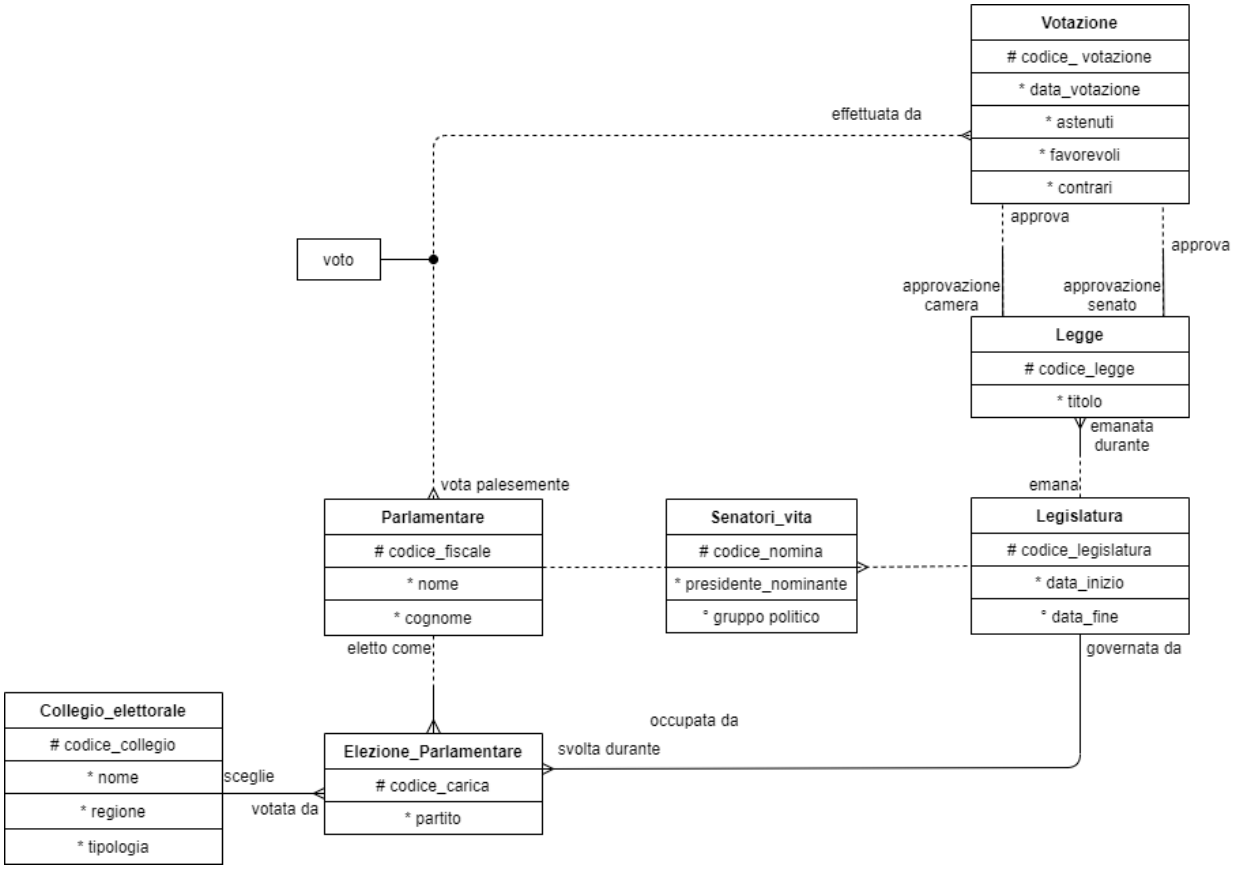
Per accedere alle informazioni da remoto ho optato per la realizzazione di un webserver mediante Node JS, runtime di Javascript che permette di realizzare applicazioni lato server codificate in javascript.

Infine la progettazione di un sito web (html, css, js) consente, grazie alla interfaccia grafica, l'accesso ai dati anche a persone meno esperte nel campo informatico.

Progettazione database - B

Schema concettuale

Per aiutarsi nella modellazione di una base di dati è fondamentale riordinare le richieste del cliente e sviluppare un diagramma er in grado di soddisfarle, se possibile, nella loro totalità.



Questo in figura è il diagramma ER ristrutturato che ho prodotto per soddisfare le richieste della traccia, di seguito andrò a giustificare alcune delle scelte fatte nella realizzazione di esso.

La prima relazione che analizzerò è Legislatura, in particolare di questa tabella ciò che voglio sottolineare è l'opzionalità del campo data_fine. Alla creazione di una nuova legislatura infatti non si potrà conoscere la data in cui andrà a terminare in quanto essa potrebbe essere anticipata da imprevisti, come nel caso della XV legislatura durata solo 2 anni.

Successivamente, seguendo la traccia, incontriamo la relazione Parlamentare. Notare come all'interno di essa non si trovano né il campo partito né quello per distinguere senatori da deputati. Il campo partito penso sia infatti più adatto all'interno della relazione Elezione_Parlamentare, è infatti possibile che un parlamentare cambi partito durante la sua carriera politica.
La distinzione tra deputati e senatori avverrà invece controllando la tipologia del collegio nel quale sono stati eletti; salvo l'eccezione dei senatori a vita che saranno riassunti nella relazione apposita.
Per quanto riguarda la relazione Elezione_Parlamentare un'aspetto su cui vorrei concentrarmi è la sua associazione alla relazione Parlamentare: è infatti obbligatorio per una Elezione essere associata ad un Parlamentare ma un Parlamentare potrebbe non essere mai stato eletto in quanto nominato a vita da un presidente. Da notare come gruppo_politico nella relazione Senatori_Vita sia opzionale, alcuni senatori a vita possono infatti non far parte di nemmeno uno schieramento.
Un'altra parte importante da realizzare era quella relativa alle leggi, si richiedeva di tenere traccia delle leggi emanate, della relativa legislatura e dei dati relativi alle approvazioni di camera e senato. La cosa che mi sento di dover spiegare riguardo questa parte di diagramma è la non obbligatorietà da parte di una votazione di risultare in una legge. La traccia chiedeva infatti di riportare solo le leggi approvate, leggendo solo questo punto, sembra logico che anche ogni votazione riportata nel database debba associarsi ad una legge. Ciò che mi ha spinto a rendere queste associazioni opzionali è stata però la possibilità della presenza di voti palesi; è infatti rilevante poter controllare come hanno votato i parlamentari una determinata legge, anche nel caso essa non venisse approvata.

Schema logico

Lo schema logico prodotto è composto da 8 tabelle:

Parlamentare(codice_parlamentare, nome, cognome)
Elezione_Parlamentare(codice_elezione, partito, *codice_parlamentare*, *codice_legislatura*, *codice_collegio*)
Collegio_Elettorale(codice_collegio, nome, regione, tipologia)
Legislatura(codice_legislatura, data_inizio, data_fine)
Senatori_Vita(codice_nomina, presidente_nominante, gruppo_politico, *parlamentare_nominato*, *codice_legislatura*)
Legge(codice_legge, titolo, *approvazione_camera*, *approvazione_senato*, *legislatura_emanante*)
Votazione(codice_votazione, data_votazione, astenuti, favorevoli, contrari)
Voto_Palese(*codice_parlamentare*, *codice_votazione*, voto)

Progettazione sito web - B

Divisione del sito

il sito web sarà suddiviso in 5 parti principali: home, deputati , senatori, leggi, avanzate.

Home:
pagina iniziale del sito dalla quale si potrà navigare semplicemente nelle altre sezioni, design minimale con 4 semplici bottoni per la navigazione, un grande titolo in centro e un'immagine che ricopre interamente lo sfondo.

Deputati:
sezione adibita alla visualizzazione dei deputati. In essa sarà disponibile una semplice interfaccia composta da due selezioni che permetteranno una ricerca attraverso legislatura e collegio di elezione.

Senatori:
sezione molto simile a quella dei deputati con le stesse funzionalità ma adibita ai senatori.

Leggi:
sezione che si occupa di mostrare le leggi approvate, vi sarà la possibilità di effettuare una ricerca per legislatura emanante.

Avanzate:
questa sezione è dedicata maggiormente a programmatori e data analisti. Essa infatti rende disponibile la possibilità di interrogare il database scrivendo personalmente le query che si desiderano; sarà ovviamente fondamentale consentire solamente le interrogazioni negando tentativi di modifica, aggiunta o cancellazione di dati e tabelle.

Tutte le sezioni avranno disponibile un tasto in alto a destra che permetterà di tornare rapidamente alla home.
In tutte le ricerche inoltre lasciare i valori di default equivarrà alla restituzione dei dati senza alcun filtraggio.

Istruzioni DDL - C

```
drop database if exists legislature_repubblica;
create database legislature_repubblica;
use legislature_repubblica;

create table Collegio_elettorale (
    codice_collegio int primary key auto_increment,
    nome varchar(30) not null,
    regione varchar(30) not null,
    tipologia varchar(30) not null,
    CHECK (tipologia = "deputati" OR tipologia = "senatori")
);

create table Legislatura (
    codice_legislatura int primary key auto_increment,
    data_inizio date not null,
    data_fine date
);

create table Votazione (
    codice_votazione int primary key auto_increment,
    data_votazione date not null,
    astenuti int not null,
    favorevoli int not null,
    contrari int not null
);

create table Legge (
    codice_legge int primary key auto_increment,
    titolo varchar(150) not null,
    approvazione_camera int not null,
    approvazione_senato int not null,
    legislatura_emanante int not null,
    constraint FK_Legge_Votazione_Camera
    foreign key (approvazione_camera) references Votazione(codice_votazione),
    constraint FK_Legge_Votazione_Senato
    foreign key (approvazione_senato) references Votazione(codice_votazione),
    constraint FK_Legge_Legislatura
    foreign key (legislatura_emanante) references Legislatura(codice_legislatura)
);
```

```sql
create table Parlamentare (
    codice_parlamentare int primary key auto_increment,
    nome varchar(30) not null,
    cognome varchar(30) not null
);

create table Voto_palese (
    codice_parlamentare int,
    codice_votazione int,
    voto varchar(20) not null,
    CHECK (voto IN ("pro","contro","astenuto","assente")),
    constraint FK_Voto_palese_Parlamentare
    foreign key (codice_parlamentare) references Parlamentare(codice_parlamentare),
    constraint FK_Voto_palese_Votazione
    foreign key (codice_votazione) references Votazione(codice_votazione),
    primary key(codice_parlamentare, codice_votazione)
);

create table Senatori_vita(
    codice_nomina int primary key auto_increment,
    presidente_nominante varchar(25) not null,
    gruppo_politico varchar(35),
    parlamentare_nominato int not null,
    codice_legislatura int not null,
    constraint FK_Senatori_vita_Parlamentare
    foreign key (parlamentare_nominato) references Parlamentare(codice_parlamentare),
    constraint FK_Senatori_vita_Legislatura
    foreign key (codice_legislatura) references Legislatura(codice_legislatura)
);

create table Elezione_Parlamentare(
    codice_elezione int primary key auto_increment,
    partito varchar(35) not null,
    codice_parlamentare int not null,
    codice_legislatura int not null,
    codice_collegio int not null,
    constraint FK_Elezione_Parlamentare_Parlamentare
    foreign key (codice_parlamentare) references Parlamentare(codice_parlamentare),
    constraint FK_Elezione_Parlamentare_Legislatura
    foreign key (codice_legislatura) references Legislatura(codice_legislatura),
    constraint FK_Elezione_Parlamentare_Collegio_elettorale
    foreign key (codice_collegio) references Collegio_elettorale(codice_collegio)
);
```

Tra gli allegati troverete tale codice nel documento "**DDL.sql**" all'interno della cartella **"Database"**, sempre all'interno di questa si troverà il documento "**creazione_insermento.sql**" il quale una volta eseguito creerà il database, vi inserirà i dati e creerà le viste.

Impianto strutturale - D

Client

La realizzazione del sito web è avvenuta tramite l'utilizzo dei linguaggi HTML, css e javascript.
Il sito è composto da una singola pagina divisa in 5 sezioni spiegate in precedenza nel punto "Progettazione sito web"; tra gli allegati troverete una cartella chiamata "**client**" che conterrà il codice relativo alla realizzazione di esso.

Web Server

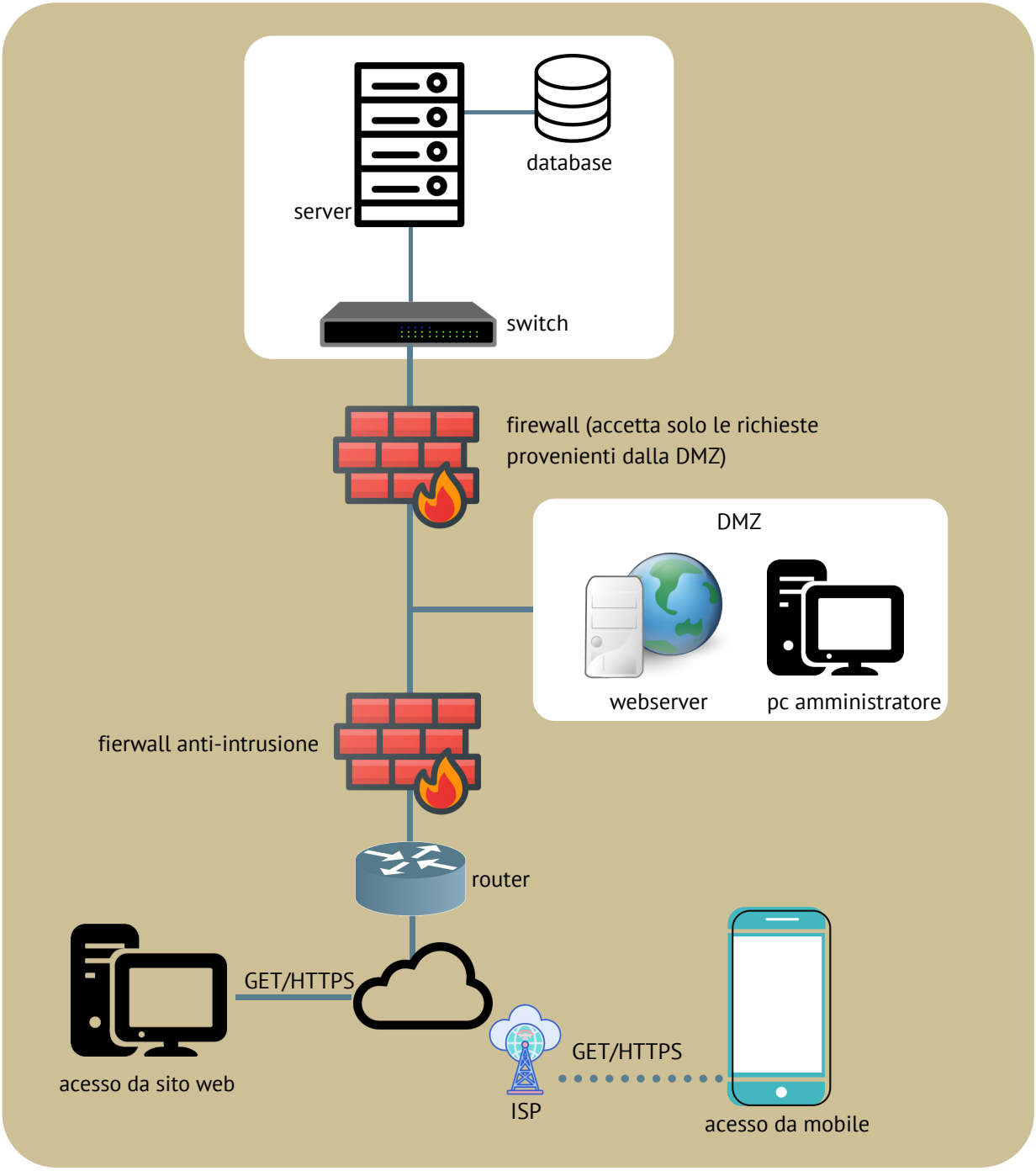
Il webserver è stato realizzato tramite l'utilizzo di **Node JS**, una runtime di javascript che permette di scrivere codice utilizzando questo linguaggio anche per il lato server.
Il suo compito è quello di interrogare il database e restituire ai client le risposte alle relative query.
Le librerie che ho deciso di utilizzare sono 2:

- **websocket** (per la connessione con il client)
- **mysql** (per l'interrogazione del database)

questa parte sarà presente negli allegati all'interno della cartella **"webserver".** Sarà necessario lanciare il documento **"index.js"** da terminale posizionandosi nella cartella **"webserver"** e lanciando il comando **node index.**

Rete - E

Rappresentazione grafica



Spiegazione scelte intraprese

Nonostante i dati siano di dominio pubblico sarà comunque importante evitare intrusioni che potrebbero portare alla modifica, aggiunta o cancellazione indesiderata di dati.

Per questo motivo sarà necessario porre il server all'interno di una **stanza sicura**, possibilmente con un solo accesso sorvegliato sempre dal personale di sicurezza. Tale stanza dovrà inoltre avere un collegamento alla fibra ottica, una temperatura inferiore ai 20 e un gruppo di continuità che fornisca corrente al sistema in caso di blackout (evitando la perdita dei dati).

Sarebbe inoltre utile utilizzare la tecnica **RAID 5**, essa infatti consentirebbe un accesso più rapido ai dati, permettendo di soddisfare meglio le richieste della grande quantità di utenti che si presume potranno richiedere un servizio di questo tipo. In caso di danneggiamento dei dati, grazie questa tecnica, sarà inoltre possibile ricostruirli. Tutti questi vantaggi hanno però un costo maggiore dovuto ad un grande spazio di archiviazione richiesto. Nonostante ciò credo che lo stato italiano possa permettersi di adoperare questa tecnica per migliorare il proprio servizio. Un altro svantaggio sarà quello di un maggior tempo necessario alla scrittura dei dati a causa del calcolo della parità. Tuttavia, siccome i dati gestiti necessitano di aggiornamenti abbastanza di rado, questo non sarà un grande problema.

La zona server sarà protetta da un firewall che consentirà l'accesso solo ai pacchetti provenienti dalla Demilitarized zone. All'interno della DMZ inseriremo il webserver (codificato con node js) e il pc dell'amministratore del database.
Quest'ultimo sarà l'unico che potrà modificare i dati e la struttura del database; per permettere ciò è necessario creare all'interno del server un apposito utente identificato dall'indirizzo IP della macchina del nostro admin a cui garantiremo tutti i privilegi.
In questo modo avremmo due fattori di sicurezza:

- username e password
- possibilità di accesso solo da un determinato pc all'interno della DMZ

```
CREATE USER 'admin'@'ip_admin' IDENTIFIED BY 'password';
GRANT ALL PRIVILEGES ON *.* TO 'user'@'ip_admin';
ALTER USER 'user'@'ip_admin' IDENTIFIED WITH mysql_native_password BY 'password';
FLUSH PRIVILEGES;
```

istruzioni necessarie alla creazione dell'utente amministratore

Per gli stessi motivi di sicurezza sarà anche necessario permettere al webserver la sola interrogazione del database revocando i privilegi relativi alle modifiche di esso.

```
CREATE USER 'user'@'%' IDENTIFIED BY 'user';
REVOKE ALL privileges ON *.* FROM 'user'@'%';
GRANT SELECT ON legislature_repubblica.* TO 'user'@'%';
ALTER USER 'user'@'%' IDENTIFIED WITH mysql_native_password BY 'user';
FLUSH PRIVILEGES;
```

istruzioni necessarie alla revoca dei privilegi di modifica al webserver

Per proteggere la Demilitarized zone sarà inoltre usato un firewall anti intrusione con il compito di analizzare i comandi impartiti dagli utenti e scoprire possibili tentativi di modifica alla base di dati. Esso verrà ovviamente posto tra il router e la DMZ.

Si potrà accedere al servizio sia da computer che da mobile grazie ad una pagina web responsive. Siccome i dati forniti saranno tutti di dominio pubblico non sarà necessaria la crittografia di essi e nemmeno l'autenticazione da parte degli utenti.