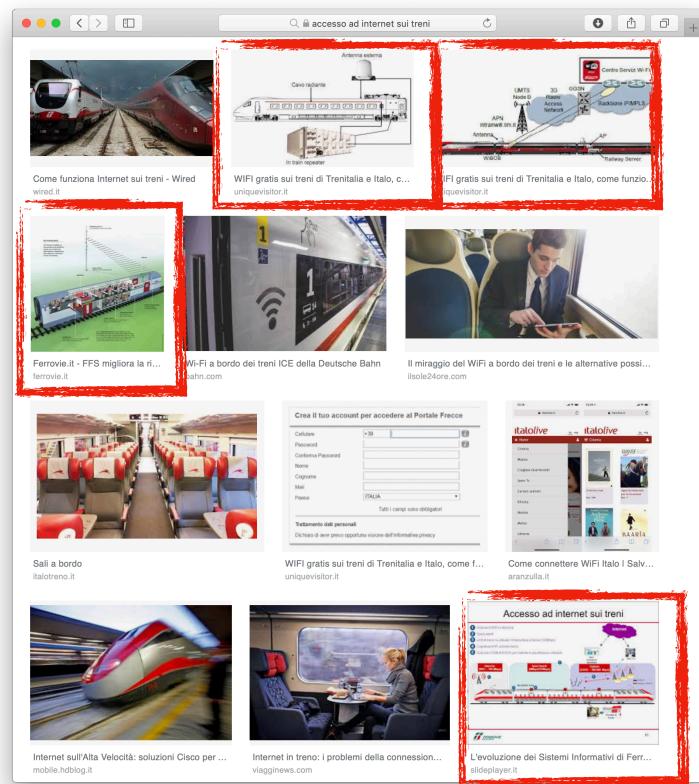


Simulazione 2 aprile 2019

informatica sistemi e reti

Questa seconda traccia ministeriale, disponibile al seguente link [seconda simulazione](#), ci presenta una compagnia ferroviaria *EasyTrain* che ha bisogno di un sistema informatico per la prenotazione online di un determinato viaggio e poi di un sistema per il controllo dei biglietti e anche per la visualizzazione in streaming di film per gli utenti in viaggio. La traccia, sia a noi insegnanti che agli studenti, è parsa molto più semplice e fattibile rispetto alla prima simulazione, anche perché descrive un sistema esistente. Sia per Italo che per Trenitalia, i biglietti si possono acquistare online e a bordo il personale, dotato semplicemente di uno smartphone, inquadra il QR-code o inserisce il codice inviato al passeggero via SMS per controllare la prenotazione. Sui treni ad alta velocità esiste una rete wifi free accessibile ad un utente registrato al portale della società dei trasporti che mette a disposizione film, quotidiani e molte informazioni online. La soluzione della infrastruttura informatica era facilmente reperibile in rete googlando le immagini per ‘*Accesso ad internet sui treni*’ come mostrato nella figura qui accanto dove vengono riportate ed evidenziate le figure, anche fornite dal sito di ferrovie dello stato, che descrivono il cablaggio dei convogli ferroviari.



Per quanto riguarda il sistema di streaming di film, anche in modalità offline, subito mi è venuto in mente una replica dei sistemi da anni in uso sugli aerei che connettono anche dei display dedicati sui sedili dei passeggeri ma, negli ultimi anni, anche i device dei passeggeri.

La Lufthansa ci fornisce una immagine che illustra un sistema di ‘*Audio Video on demand on Board*’ esistente e funzionante (link: https://www.tomshw.it/files/2014/02/immagini_contenuti/53556/boardconnect-graphic.jpg).

Anche per questa seconda simulazione, la prima parte si divide in **tre punti**.

Il **primo punto** riguarda l'infrastruttura tecnologica, come la volta precedente, con la richiesta specifica di due viste differenti: l'architettura informatica nel punto a) e nel punto b) c'è una richiesta specifica per un sistema di streaming che sia in grado di garantire la visione dei film anche in assenza di rete.

Il **secondo punto** rappresenta una novità in complessità, perché la richiesta di progettazione di una parte di database è veramente molto semplice, probabilmente la più semplice nelle tracce degli ultimi anni.

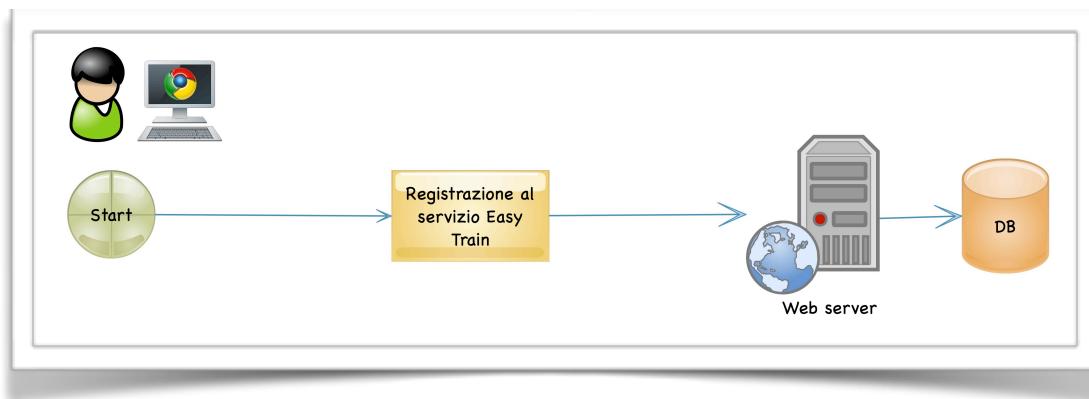
Il **terzo punto** richiede tre interrogazioni SQL di bassa/media difficoltà.

La parte di programmazione, HTML e PHP, viene riportata in una delle quattro domande della seconda parte del compito con un'altra domanda di informatica molto semplice che chiede di risalire ad un diagramma ER avendo uno schema logico.

PRIMA PARTE

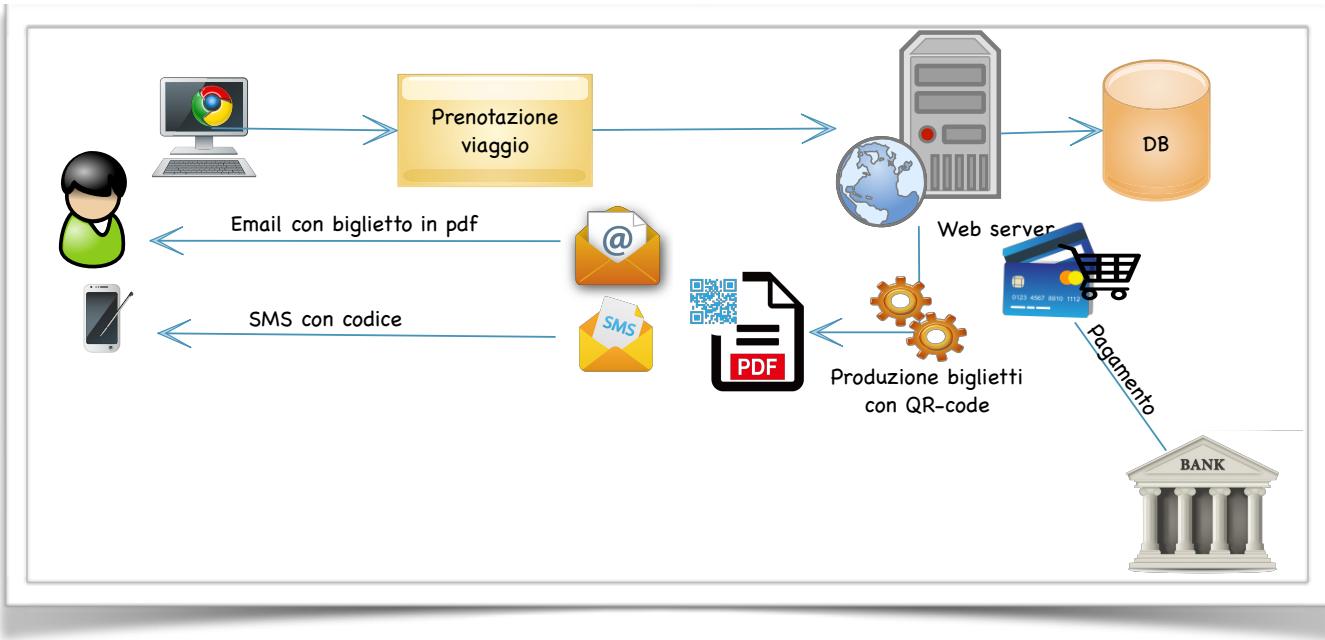
1. Quesito 1 - Il progetto, anche mediante rappresentazioni grafiche, dell'infrastruttura tecnologica ed informatica necessaria a gestire il servizio nel suo complesso

Anche se non richiesto dalla traccia, preferisco sempre chiarire il funzionamento del sistema informativo da realizzare partendo dal disegno dei processi e delle entità che ne sono coinvolte.



Essendo una registrazione via web, capiamo che ci sarà il nostro portale su un web server dedicato con un database dove verranno memorizzati i dati dell'utente.

La seconda interazione con il sistema *EasyTrain* è la ricerca, prenotazione e quindi acquisto di un titolo di viaggio. In risposta il sistema, dopo aver effettuato il pagamento, invia un pdf via email con il QR-code del biglietto e un codice via SMS.



L'utente via browser si collega al portale web dell'*EasyTrain*, cerca e prenota un viaggio e, effettuato il pagamento, riceve il biglietto in formato PDF via email e anche un codice via SMS.

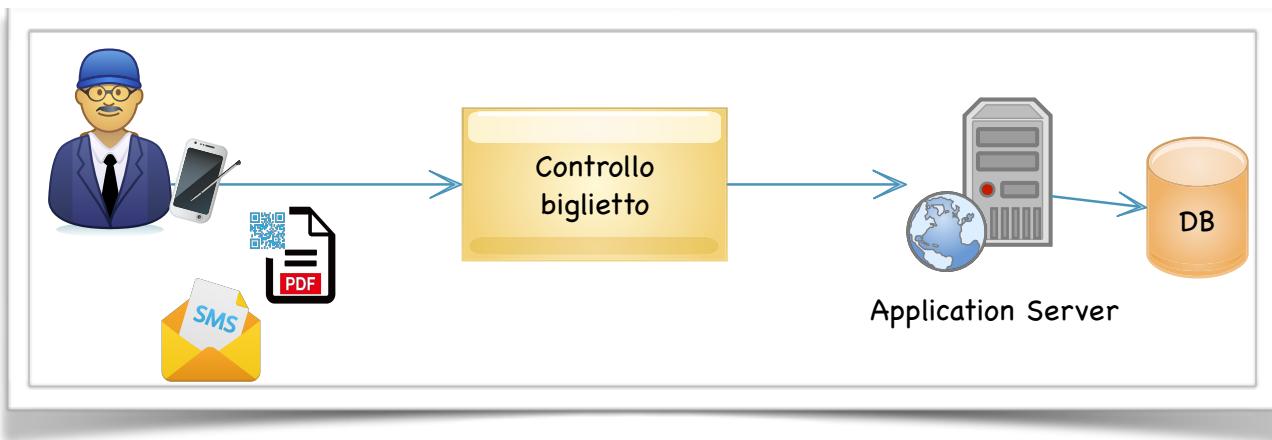
Fino a questo punto abbiamo una architettura informatica classica per una tipologia di servizio web con un modulo e-commerce. In aggiunta c'è la prenotazione di un posto su un convoglio per un determinato tragitto e la produzione di un pdf con QR-code o SMS su telefonino dell'utente. Moduli e servizi molto semplici da realizzare anche con prodotti open source.

Una seconda parte della traccia recita quanto segue:

Il personale di servizio sul treno, ad ogni stazione, effettua la verifica dei biglietti dei viaggiatori saliti a bordo, confermando la presenza di ciascun viaggiatore ed il posto occupato. La verifica di un biglietto avviene online tramite una applicazione su dispositivi mobili in dotazione al personale; l'applicazione consente di acquisire i dati mediante lettura ottica del QR code o, in mancanza, tramite digitazione del codice PU.

Questa parte rappresenta tutta un'altra parte del sistema ed è il controllo dei biglietti alla stazione prima di salire sul treno o non appena l'utente si è accomodato, non è chiaro anche se influente. In questo caso entrano dei nuovi attori, che saranno i controllori di viaggio, e un'app su uno smart device per scansionare il QR-code o il codice ricevuto via SMS.

L'app presente sullo smart device, nel momento in cui il convoglio è fermo ad una stazione, si collega ad un servizio esposto da un application server per una interrogazione sul database che quel determinato codice corrisponda effettivamente ad un utente registrato che ha prenotato un posto per quel treno in quel giorno.



Il fatto di specificare che il controllo si effettua in una stazione credo sia per eliminare la complicazione dell'assenza di rete durante il viaggio. Molti miei studenti hanno previsto anche il controllo dei biglietti in movimento e in assenza di rete con un meccanismo di replica sull'app mobile della mappatura del treno posti-codici-utenti. In effetti non era una cosa complicata da prevedere come ipotesi aggiuntiva o evoluzione del sistema.

Ultimo scenario è un servizio messo a disposizione dalla *EasyTrain* per l'utente:

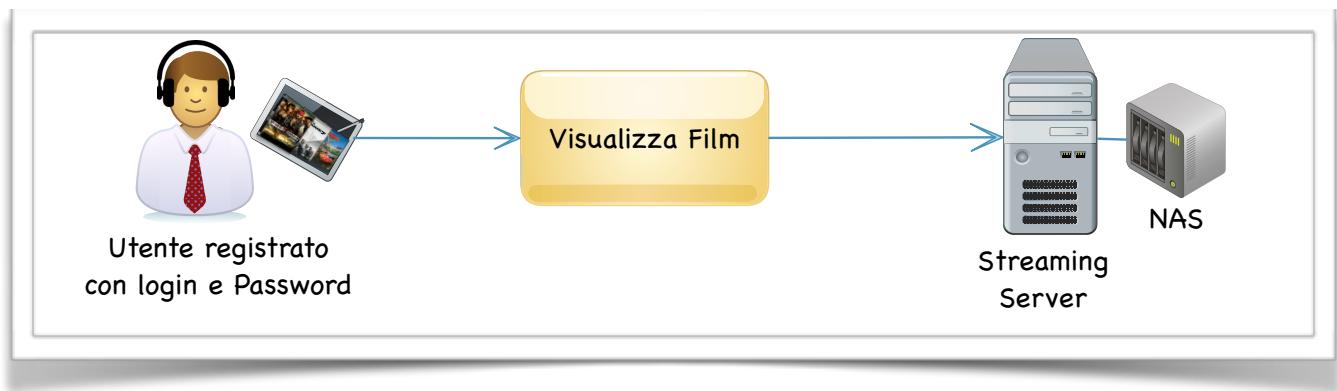
Per rendere più confortevole il viaggio, la compagnia *EasyTrain* fornisce su tutte le carrozze un servizio di wifi gratuito, a cui i viaggiatori possono accedere attraverso le stesse credenziali di accesso al portale di acquisto dei biglietti.
EasyTrain mette anche a disposizione dei viaggiatori un catalogo, frequentemente aggiornato, di una trentina di film, visualizzabili sul dispositivo mobile del viaggiatore stesso. Ciascun film in catalogo è

L'attore torna ad essere il nostro utente che, dopo una prima connessione via utente e password ai sistemi wifi del treno, si collega ad un catalogo di film messi a disposizione gratuitamente per la visualizzazione.



Per collegarsi alla wifi, l'*EasyTrain* avrà bisogno di dotarsi di un **Captive Portal**. Ricordiamo che un **captive portal** è una speciale pagina web che viene visualizzata al tentativo di connessione ad una WiFi che appare come pubblica.

Questo servirà agli utenti per collegarsi alla WiFi ed accedere ad Internet autenticandosi con le credenziali inserite nel momento in cui ci si è registrati al portale della *EasyTrain*.



Rifacendoci alla struttura descritta in prima pagina per il sistema in dotazione agli aerei, per non avere problemi di visualizzazione del film, l'utente si collegherà un piccolo streaming server, supportato da un NAS situato magari in testa al convoglio, datato di un piccolo web server per la scelta del catalogo dei film e poi sarà in grado di trasmettere, senza interruzioni o problemi di collegamento, il video anche in alta risoluzione.

Trenta film sono un numero esiguo di video da mantenere in locale. Il problema magari si potrebbe verificare per larghezza di banda necessaria anche locale nel caso in cui il treno sia completamente occupato e un buon numero di viaggiatori decida di usufruire del servizio di visualizzazione film. Su un Frecciarossa si contano anche 450 posti o più ma dovremo fare un calcolo che la maggior parte degli utenti preferisce lavorare, navigare o leggere un libro piuttosto che vedere un film, comunque dovremo tener presente questa considerazione nella scelta dei dispositivi di rete.

2. Quesito 1 - punto a) le modalità di comunicazione tra le varie componenti, relativamente alle operazioni di validazione dei biglietti sul treno e di accesso alla rete tramite credenziali da parte dei viaggiatori, descrivendo canali, dispositivi, protocolli e servizi di rete e motivando le scelte effettuate;

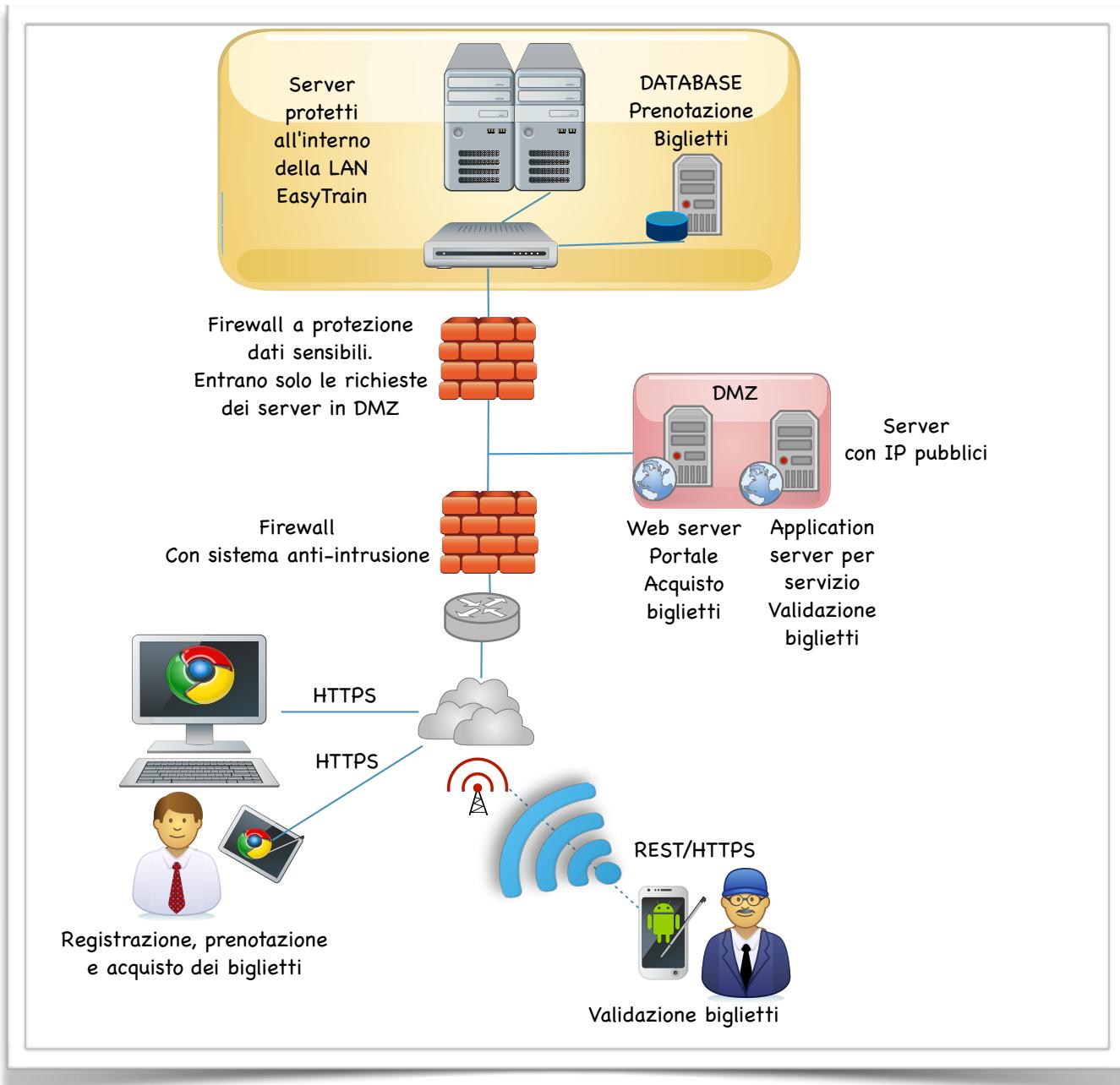
Ora, finalmente, arriviamo a rispondere al punto a) del primo quesito.

Nel punto a) viene richiesto di diagrammare l'infrastruttura per:

1. Validazione dei biglietti sul treno (*app su smart device mobile*)
2. Accesso alla rete tramite credenziali (*captive portal*)

Nel seguente diagramma illustriamo i canali, i dispositivi, i protocolli e i servizi che entrano in gioco per la validazione dei biglietti fermi ad una stazione quindi con il presupposto di essere coperti da rete e di collegarci con il sistema di Prenotazione/acquisto/verifica biglietti centrale.

Potremo anche limitarci allo schema della validazione del biglietto, una app, magari su un sistema open source android, che si collega ad un web-service di validazione posto-biglietto-utente.



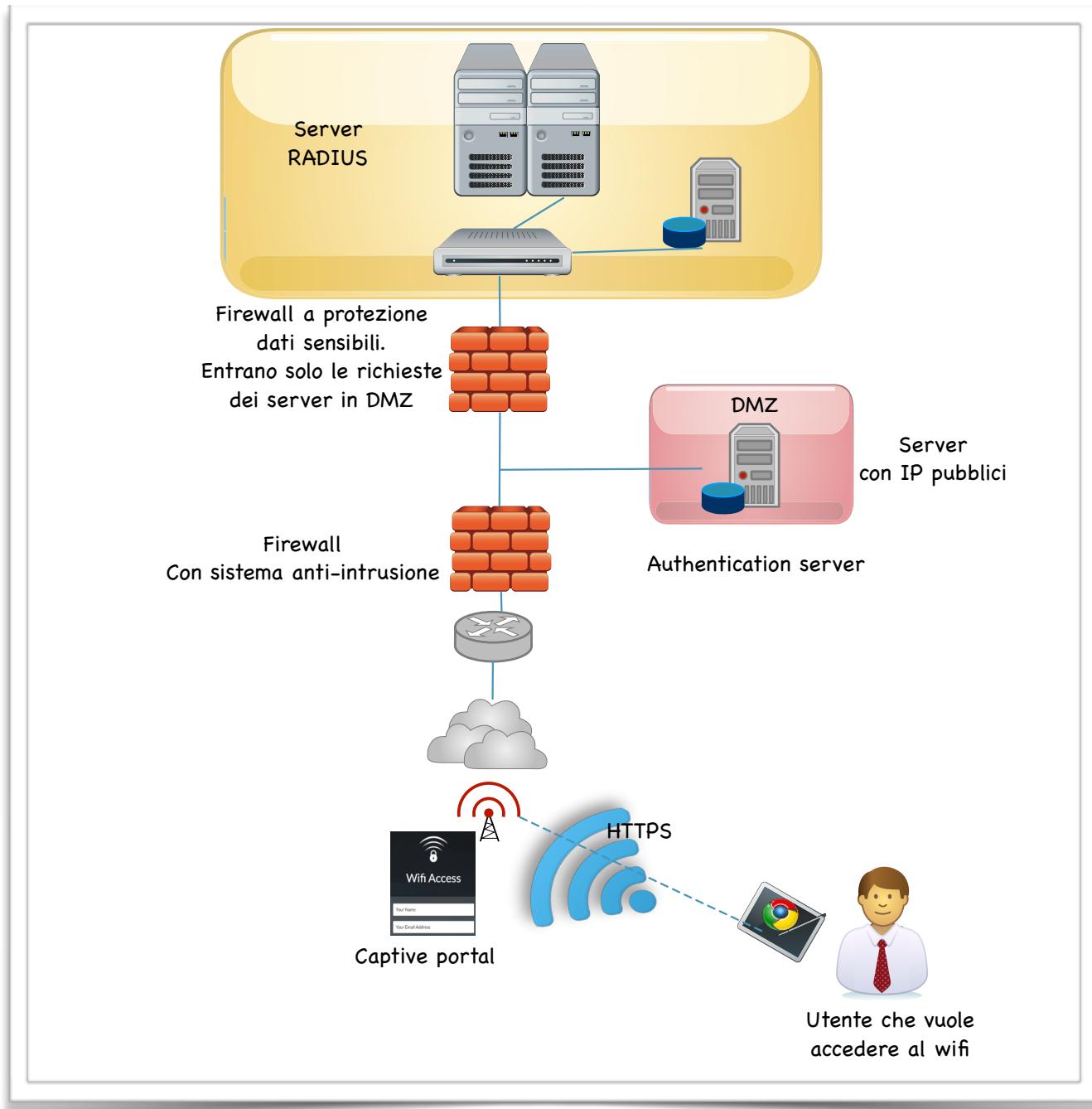
Visto che, prima di farsi validare il biglietto, un utente ha dovuto necessariamente effettuare un acquisto online, ho aggiunto un web server che contiene il portale della *EasyTrain* accanto all'application server che fornisce il servizio di validazione anche perché condivideranno il database.

Non viene richiesto, per cui non è stato aggiunto, il plugin per l'acquisto dei biglietti con collegamento ai provider delle transazioni per le carte di credito.

Il secondo diagramma chiede di disegnare il funzionamento di un captive portal, ovvero un utente si collega in rete e su un browser gli compare una pagina i cui inserire le credenziali.

Per questo tipo di architettura si deve ipotizzare un server RADIUS all'interno dell'*EasyTrain* con un server per l'autenticazione in DMZ. Ricordiamo, copiando la definizione da Wikipedia, che cosa è un server RADIUS:

RADIUS (Remote Authentication Dial-In User Service), in informatica e telecomunicazioni, è un protocollo AAA (*authentication, authorization, accounting*) utilizzato in applicazioni di accesso alle reti o di mobilità IP. Quindi con server RADIUS intendiamo un qualsiasi server di mercato che supporti questo protocollo.

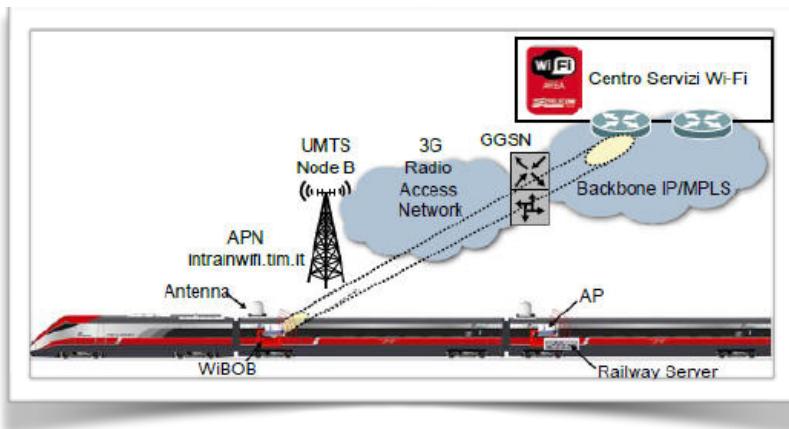


Ovviamente i server, rispetto a quelli illustrati precedentemente (authentication server/web server/application server in DMZ e server RADIUS/database server in LAN interna), possono anche convivere in un'unica macchina, l'importante è avere chiaro la differenza tra i vari servizi forniti.

Per accedere alla WiFi dobbiamo necessariamente essere coperti da rete perché il sistema di autenticazione/autorizzazione è centralizzato, sarebbe impensabile, oltre che inutile, dover replicare il database con gli utenti all'interno dei singoli treni.

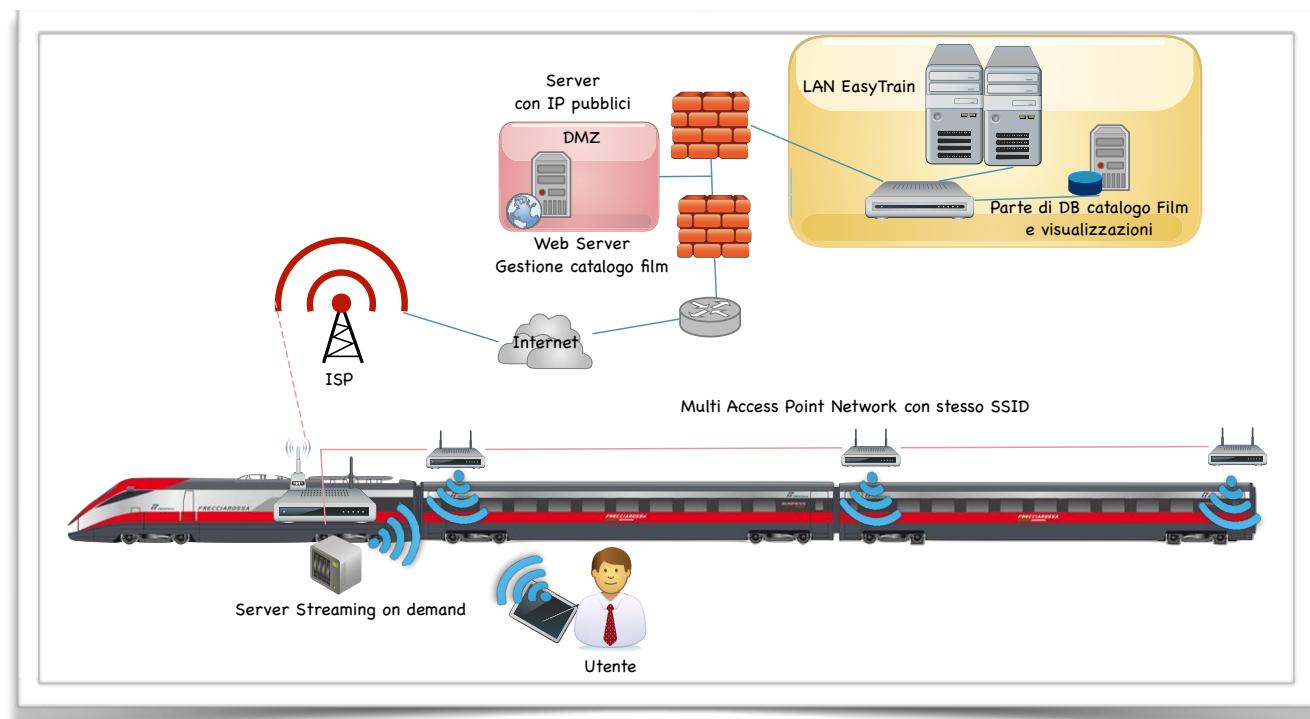
Quesito 1 - punto b) le soluzioni hardware e software per garantire una visione fluida e continuativa dei film sui dispositivi mobili dei viaggiatori indipendentemente dalle condizioni sopra esposte che influiscono sulla qualità della connessione ad Internet.

La soluzione hardware si basa su una infrastruttura di comunicazione esistente attualmente sui vari convogli italiani descritti nella seguente immagine (link: <https://www.uniquevisitor.it/images/struttura-wifi-carrozza.jpg>)



Un'antenna collegata ad un access point principale sulla prima carrozza del treno si connette ad un gestore telefonico (tim o vodafone ad esempio) e, attraverso un cavo radiante, collega i vari access point lungo il treno che forniranno ad ogni carrozza la rete wifi.

Nel seguente disegno ho aggiunto il sistema di streaming per i film che portato in locale, come per gli aerei, diventa assolutamente indipendente dalla fluttuazione della rete mobile che, in alcuni tratti montani o gallerie, è completamente assente. Esistono diversi software di streaming gratuiti, ad esempio Air Playit. Una qualche interazione con il sistema centrale, per il catalogo o per memorizzare i gusti dell'utente o quali film ha visualizzato, deve essere previsto nel momento in cui il convoglio è in rete.



2. Quesito numero 2- il database

Il database, richiesto al punto 2, fortunatamente è solo una piccola parte del sistema. Dover modellare tutto il sistema della *EasyTrain*, con le prenotazione dei viaggi, il biglietto ed i treni sarebbe stato molto più complicato per gli studenti.

il progetto della porzione della basi di dati per la gestione del catalogo dei film e della loro fruizione da parte dei viaggiatori: si richiede in particolare il modello concettuale e il corrispondente modello logico.

Rileggiamo colorando la porzione di traccia che parla del catalogo dei film. Evidenzio in giallo quelle che diventeranno le nostre entità e gli eventuali attributi in verde.

EasyTrain mette anche a disposizione dei viaggiatori un catalogo, frequentemente aggiornato, di una trentina di **film**, visualizzabili sul dispositivo mobile del **viaggiatore** stesso. Ciascun film in catalogo è descritto da una scheda che, oltre al **titolo**, riassume le caratteristiche del film quali **genere**, **durata**, **attori** principali, breve descrizione della **trama**, **trailer**. Per aggiornare il catalogo, *EasyTrain* si basa anche sulle statistiche di visualizzazione dei film da parte dei viaggiatori.

La soluzione contiene di sicuro le seguenti entità:

UTENTE: Potremo anche chiamarlo viaggiatore. Gli attributi da inserire su questa entità sono descritti all'inizio della traccia.

FILM: Ha un codice univoco e una serie di attributi. Attenzione agli attori che, anche se elencati come attributo del film, deve diventare assolutamente una entità. Qualche studente, nella fretta, ha inserito gli attori come attributo del film.

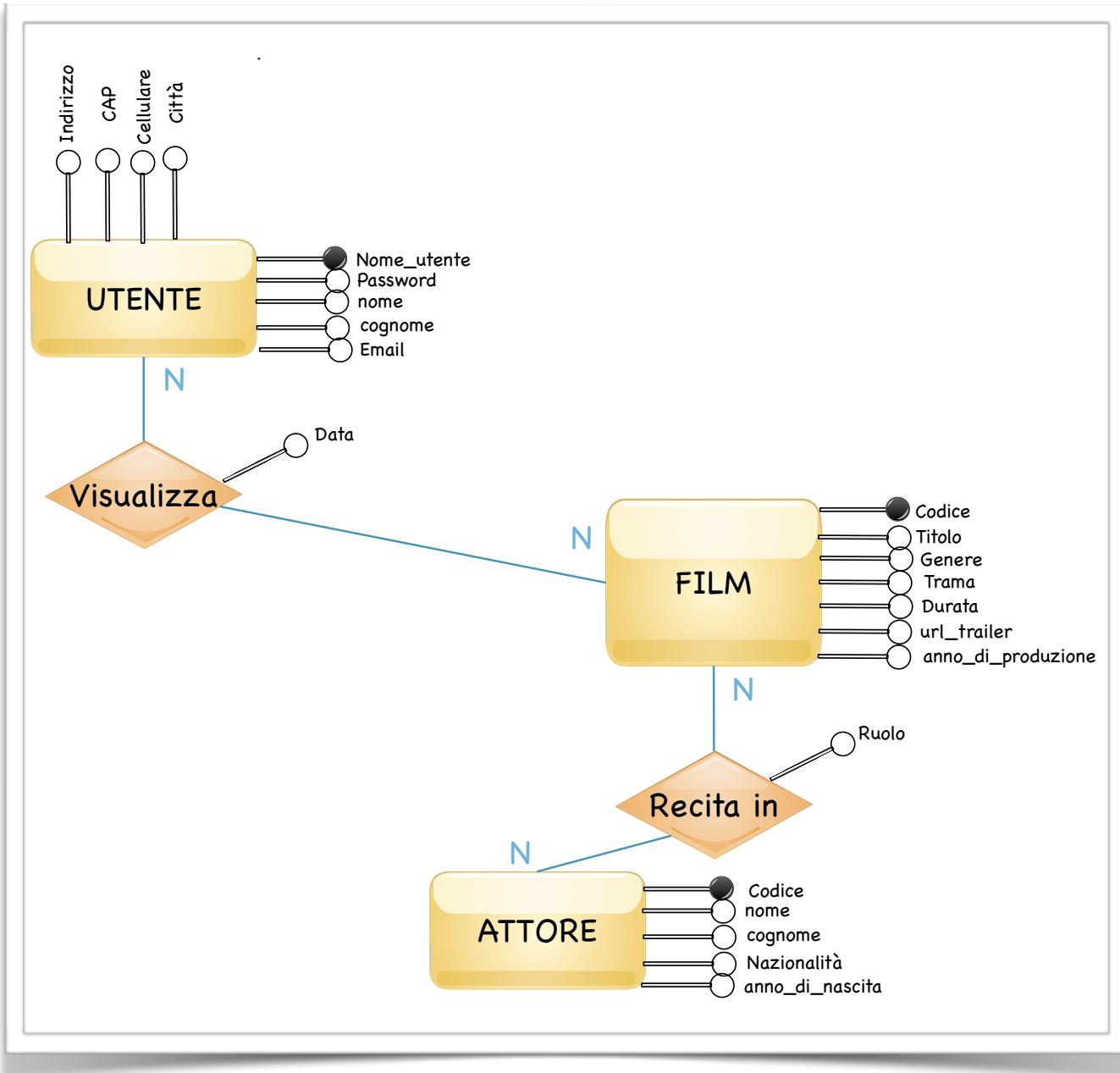
ATTORE: L'attore è una persona e ha anche pochi attributi che dovremo inserire all'interno della scheda dei film nei quali ha partecipato.

Per definire correttamente gli attributi delle entità trovate, si deve leggere la traccia della prima parte fino in fondo ed infatti nel punto 3, all'interno delle query, compaiono altri attributi da memorizzare.

la codifica in linguaggio SQL delle seguenti interrogazioni:

- a) elenco dei film in catalogo ordinati per **genere** ed **anno di produzione**;
- b) elenco in ordine alfabetico degli utenti che non hanno mai visualizzato alcun film;
- c) dato un intervallo di tempo tra due **date**, produrre il titolo che ha registrato il maggior numero di visualizzazioni.

Il diagramma ER risultante:



Relazione **visualizza** tra **UTENTE** e **FILM** multi-a-multi: un utente visualizza nell'arco del tempo più film e un singolo film può essere visti da più utenti. La relazione ha l'attributo '*data*' perché deve essere memorizzato il giorno in cui l'utente ha visualizzato il film.

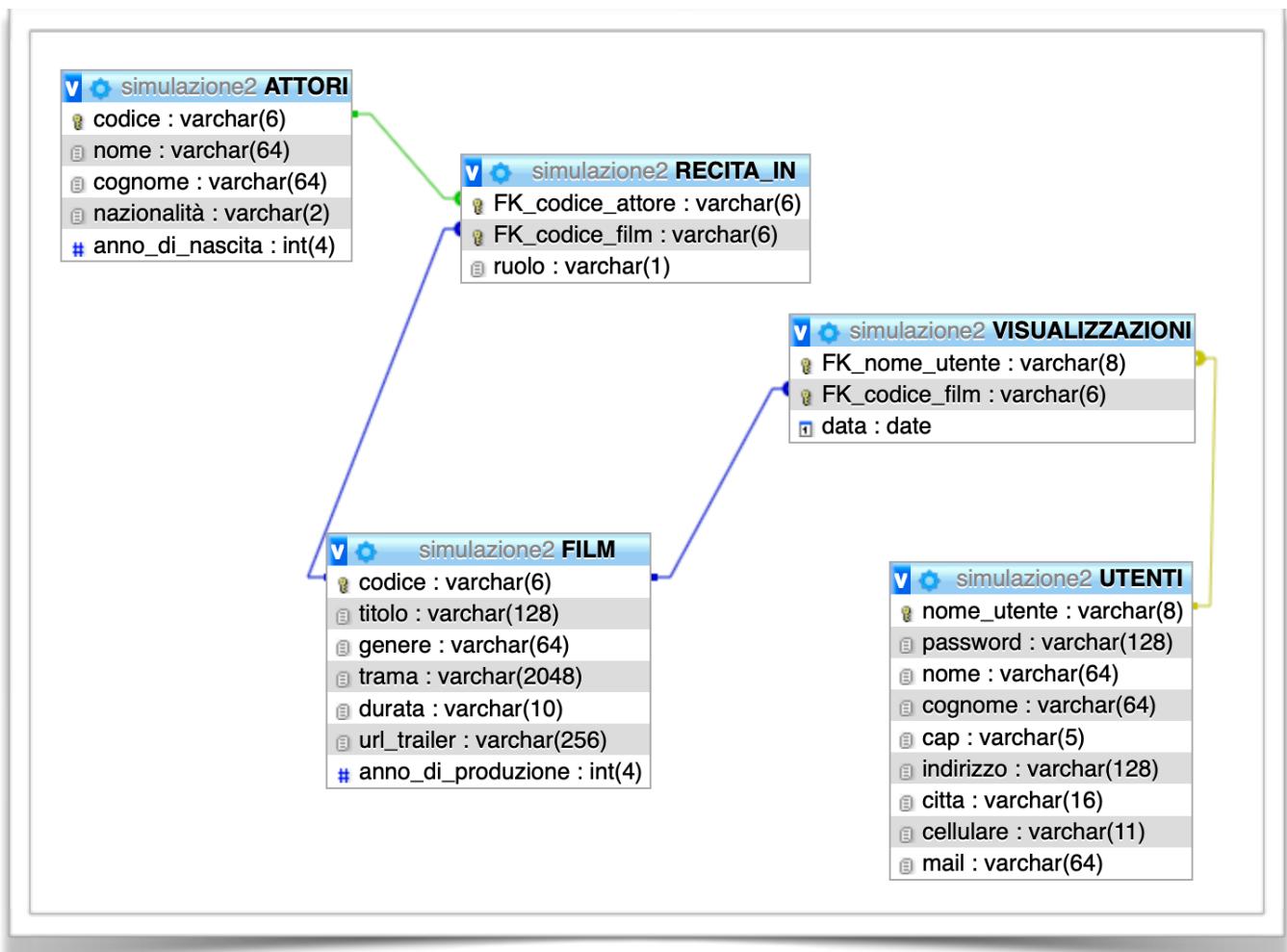
Relazione **recita in** tra **ATTORE** e **FILM** multi-a-multi: un classico della progettazione dei database. Un attore recita in più film, in un film recitano più attori. Sulla relazione si aggiunge un attributo '*ruolo*' che magari verrà visualizzato nella scheda del film come protagonista, coprotagonista o comparsa.

Modello logico

Applicando le regole di derivazione abbiamo il seguente modello logico:

UTENTI(nome_utente, password, nome, cognome, email, indirizzo, cap, città, cellulare);
FILM(codice, titolo, genere, trama, durata, url_trailer, anno_di_produzione);
ATTORI(codice, nome, cognome, nazionalità, anno_di_nascita);
VISUALIZZAZIONI(FK_nome_utente, FK_codice_film, data);
RECITA_IN(FK_codice_attore, FK_codice_film, ruolo);

Le entità diventano tabelle, prendono un nome al plurale e gli attributi diventano campi.
Le relazioni molti a molti si traducono con la creazione di una terza tabella.
La tabella VISUALIZZAZIONI si porta dietro le foreign key verso gli UTENTI e verso i FILM con l'aggiunta del campo data e la tabella RECITA_IN si porta le foreign key verso l'attore ed il film.



3. Quesito 3 - la codifica in linguaggio SQL delle seguenti interrogazioni

- a) elenco dei film in catalogo ordinati per genere ed anno di produzione;

Prima query molto facile perché su una sola tabella senza condizioni ma solo con l'ORDER BY.

```
SELECT * FROM FILM ORDER BY genere, anno_di_produzione
```

- b) elenco in ordine alfabetico degli utenti che non hanno mai visualizzato alcun film;

Diverse possibilità di eseguire questa query. Utilizzando gli insiemi (NOT IN) e select di select:

```
SELECT utenti.cognome, utenti.nome FROM `utenti` WHERE
nome_utente NOT IN
(SELECT fk_nome_utente from visualizzazioni) order by
cognome, nome
```

oppure con la left JOIN tra le tabelle utenti e visualizzazioni:

```
SELECT utenti.cognome, utenti.nome
FROM utenti LEFT JOIN VISUALIZZAZIONI ON utenti.nome_utente
= VISUALIZZAZIONI.FK_nome_utente
WHERE VISUALIZZAZIONI.FK_nome_utente is null ORDER BY
cognome, nome
```

- c) dato un intervallo di tempo tra due date, produrre il titolo che ha registrato il maggior numero di visualizzazioni.

```
SELECT titolo, COUNT(codice) as vis
FROM film, VISUALIZZAZIONI
WHERE film.codice = VISUALIZZAZIONI.FK_codice_film
and VISUALIZZAZIONI.data BETWEEN '2019-04-01' and
'2019-04-04'
GROUP BY titolo ORDER BY vis DESC LIMIT 1
```

SECONDA PARTE

Quesito I - Codice PHP

Per la seconda parte sceglieremo il primo quesito sul PHP. Le query svolte sopra verrano inserite in un codice PHP, non viene richiesta la form HTML ma solo il pezzo di codice che esegue le query.

Codice PHP per la prima query

```
<html><head>
  <title>Seconda Simulazione 2019 - prima query</title></head>
<body>
  <?
  $conn = new mysqli("localhost","root","","simulazione2");
  if ($conn->connect_error) {
    die("Connessione fallita con errore: " . $conn->connect_error);
  }
  $query = "SELECT titolo, genere, anno_di_produzione FROM FILM ORDER BY genere, anno_di_produzione";
  $result = $conn->query($query);
  $i = 0;
  if ($result->num_rows > 0) {
    echo "elenco dei film in catalogo ordinati per genere ed anno di produzione <br>";
    echo "<table border=\"1\">";
    echo "<tr><td bgcolor=\"#FFFFFF\">Titolo</td><td  bgcolor=\"#FFFFFF\">genere</td>";
    echo "<td  bgcolor=\"#FFFFFF\">Anno di produzione</td></tr>";

    while($row = $result->fetch_assoc()) {
      if($i % 2 == 0){
        echo "<tr><td bgcolor=\"#BBBBBB\">&ampnbsp".$row['titolo'];
        echo "</td><td  bgcolor=\"#BBBBBB\">&ampnbsp".$row['genere'];
        echo "</td><td  bgcolor=\"#BBBBBB\">&ampnbsp".$row['anno_di_produzione']."'</td></tr>";
      }
      else{
        echo "<tr><td bgcolor=\"#DDDDDD\">&ampnbsp".$row['titolo'];
        echo "</td><td  bgcolor=\"#DDDDDD\">&ampnbsp".$row['genere'];
        echo "</td><td  bgcolor=\"#DDDDDD\">&ampnbsp".$row['anno_di_produzione']."'</td></tr>";
      }
      $i++;
    }
  } else {echo "Nessun risultato";}
  echo "</table>";
  $conn->close();
  ?>
</body>
</html>
```

Nel browser viene visualizzato così:

elenco dei film in catalogo ordinati per genere ed anno di produzione

Titolo	genere	Anno di produzione
Titanic	drammatico	1997
Gangs of New York	drammatico	2002
The Aviator	drammatico	2004
Star Wars - Episodio 1	fantascienza	1999
Star Wars - Episodio 2	fantascienza	2002
Shutter Island	thriller	2010

Codice PHP per la seconda query

```

<html>
    <head> <title>Seconda Simulazione 2019 – seconda query</title> </head>
<body>
    <?
    $conn = new mysqli("localhost","root","","simulazione2");
    if ($conn->connect_error) {
        die("Connessione fallita con errore: " . $conn->connect_error);
    }
    $query = "SELECT utenti.cognome, utenti.nome FROM utenti WHERE nome_utente NOT IN ".
        "(SELECT fk_nome_utente from visualizzazioni) order by cognome,nome";
    $result = $conn->query($query);
    $i = 0;
    if ($result->num_rows > 0) {
        echo "<br>Elenco in ordine alfabetico degli utenti che non hanno mai visualizzato alcun film<br><br>";
        echo "<table border=\\"1\\">";
        echo "<tr><td bgcolor=\\"#FFFFFF\\>Cognome</td><td  bgcolor=\\"#FFFFFF\\>Nome</td>";
        echo "</tr>";

        while($row = $result->fetch_assoc()) {
            if($i % 2 == 0){
                echo "<tr><td bgcolor=\\"#BBBBBB\\>&ampnbsp".$row['cognome'];
                echo "</td><td  bgcolor=\\"#BBBBBB\\>&ampnbsp".$row['nome']."'</td></tr>";
            }
            else{
                echo "<tr><td bgcolor=\\"#DDDDDD\\>&ampnbsp".$row['cognome'];
                echo "</td><td  bgcolor=\\"#DDDDDD\\>&ampnbsp".$row['nome']."'</td></tr>";
            }
            $i++;
        }
    } else {
        echo "0 results";
    }
    echo "</table>";
    $conn->close();
    ?>
</body>
</html>

```

Nel browser di vedrà così:

The screenshot shows a web browser window with the URL "localhost/alter/sim2_3.php". The page title is "Elenco in ordine alfabetico degli utenti che non hanno mai visualizzato alcun film". Below the title is a table with two columns: "Cognome" and "Nome". The data in the table is:

Cognome	Nome
Bianchi	Francesco
Rossi	Maria

Codice PHP per la terza query

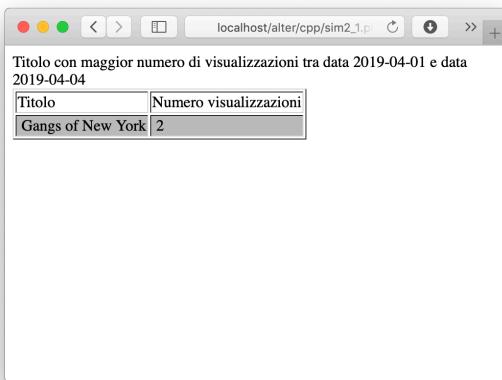
```
<html>
  <head> <title>Seconda Simulazione 2019 – terza query</title> </head>
<body>
  <? $data1=$_POST['data1'];
    $data2=$_POST['data2'];

$conn = new mysqli("localhost","root","","simulazione2");
if ($conn->connect_error) { die("Connessione fallita con errore: " . $conn->connect_error); }
$query = " SELECT titolo, COUNT(codice) as vis ".
" FROM film, VISUALIZZAZIONI ".
" WHERE film.codice = VISUALIZZAZIONI.FK_codice_film ".
" and VISUALIZZAZIONI.data BETWEEN '".$data1."' and '".$data2."' .
" GROUP BY titolo ORDER BY vis DESC LIMIT 1" ;

$result = $conn->query($query);
$i = 0;
if ($result->num_rows > 0) {
  echo "Titolo con maggior numero di visualizzazioni tra data " . $data1 ." e data ". $data2 . "<br>";
  echo "<table border='1'>";
  echo "<tr><td bgcolor=\"#FFFFFF\">Titolo</td><td  bgcolor=\"#FFFFFF\">NUmero visualizzazioni</td></tr>";

  while($row = $result->fetch_assoc()) {
    echo "<tr><td bgcolor=\"#BBBBBB\">&ampnbsp" . $row['titolo'];
    echo "</td><td  bgcolor=\"#BBBBBB\">&ampnbsp" . $row['vis']. "</tr>";
  }
} else {
  echo "Nessun risultato";
}
echo "</table>";
$conn->close();
?>
</body>
</html>
```

Nel browser si vedrà come:



Quesito III - dal logico al concettuale

III. Dato il seguente schema logico

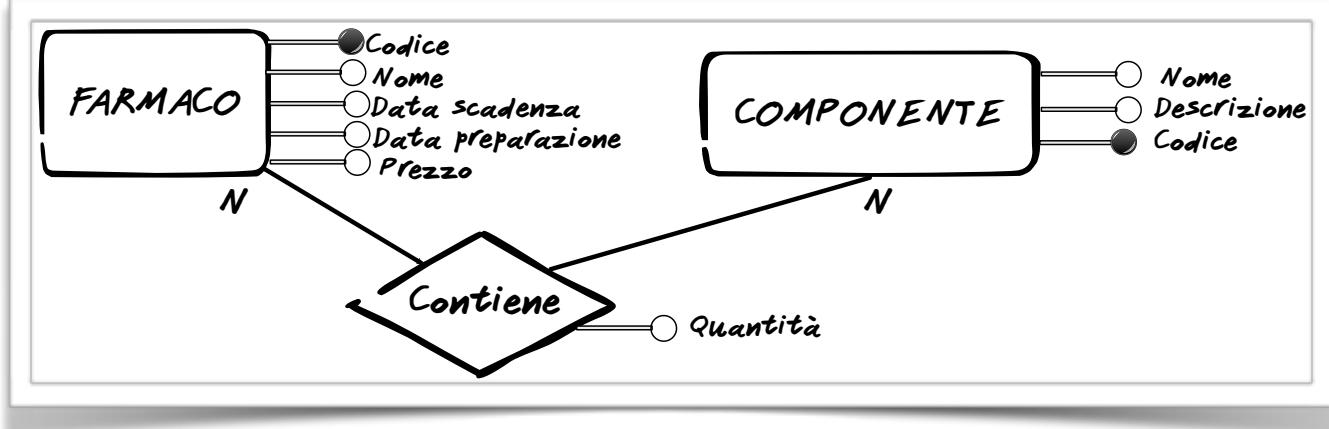
FARMACO (COD_F,NOME_F,DATA_PREPARAZIONE,DATA_SCADENZA,PREZZO)
COMPONENTE (COD_C,NOME_C,DESCRIZIONE)
CONTIENE (ID_FARMACO,ID_COMPONENTE,QUANTITA_C)

si chiede di:

- disegnare il diagramma del modello concettuale corrispondente;
- definire in linguaggio SQL il modello fisico corrispondente tenendo conto dei vincoli di integrità referenziali e/o vincoli di dominio;
- esporre il significato delle varie tipologie di vincoli che si possono riscontrare nella progettazione delle basi di dati e dei riflessi che essi hanno sulle operazioni di inserimento, aggiornamento e cancellazione.

Il terzo quesito chiede di risalire ad un modello concettuale da un modello logico.
Abbiamo una relazione molti a molti con la tabella CONTIENE che ha le foreign key verso FARMACO e verso COMPONENTE.

Ecco il modello concettuale E/R:



Il codice SQL per il modello fisico è il seguente:

```
CREATE TABLE componente (
    cod_c varchar(8) NOT NULL,
    nome_c varchar(64) NOT NULL,
    descrizione varchar(256) NOT NULL);

CREATE TABLE contiene (
    id_farmaco varchar(8) NOT NULL,
    id_componente varchar(8) NOT NULL,
    quantity_c int(11) NOT NULL);

CREATE TABLE farmaco (
    cod_f varchar(8) NOT NULL,
    nome_f varchar(64) NOT NULL,
    data_preparazione date NOT NULL,
    data_scadenza date NOT NULL,
    prezzo decimal(5,2) NOT NULL);

ALTER TABLE componente
    ADD PRIMARY KEY (cod_c);

ALTER TABLE contiene
    ADD PRIMARY KEY (id_farmaco,id_componente),
    ADD KEY id_componente (id_componente);

ALTER TABLE farmaco
    ADD PRIMARY KEY (cod_f);

ALTER TABLE contiene
    ADD CONSTRAINT contiene_ibfk_1 FOREIGN KEY (id_farmaco)
    REFERENCES farmaco (cod_f),
    ADD CONSTRAINT contiene_ibfk_2 FOREIGN KEY (id_componente)
    REFERENCES componente (cod_c);
```

La domanda sulle tipologie di vincoli poteva essere messa anche come punto a parte perché ci sarebbe molto da scrivere, ma giusto per ricordare molto brevemente:

I **vincoli di integrità** sono delle proprietà che devono essere soddisfatte dalla istanza della base di dati. Più semplicemente sono delle restrizioni sui possibili valori assunti dai dati.

Ci possono essere delle restrizioni implicite od esplicite. Tra i vincoli impliciti troviamo il **vincolo di chiave principale** che non ammette valori duplicati e deve essere necessariamente presente. Quando invece parliamo di integrità tra chiave primaria e foreign key parliamo del **vincolo di integrità referenziale** definito formalmente come segue:

Nell'ambito dei **RDBMS**, l'**integrità referenziale** è un **vincolo di integrità** di tipo interrelazionale ovvero una proprietà dei dati che, se soddisfatta, richiede che ogni valore di un attributo (colonna) di una relazione (tabella) esista come valore di un altro attributo in un'altra relazione. (fonte: Wikipedia)

Un **vincolo di tupla** o record è un **vincolo** che può essere valutato su ciascuna **tupla** indipendentemente dalle altre. Un **vincolo** definito con riferimento a singoli valori viene detto **vincolo** su valori o **vincolo di dominio** in quanto impone una restrizione sul dominio dell'attributo. (fonte: Wikipedia)