ESAME DI STATO A.S. 2020/2021

La società Media Way S.p.A vuole realizzare un sistema digitale per l'assegnazione delle frequenze dei vari operatori mediatici che partecipano ai grandi eventi.
Il candidato, formulate le opportune ipotesi aggiuntive sulle caratteristiche e la natura del problema in oggetto, sviluppi un'analisi della realtà di riferimento individuando quali devono essere le specifiche che il sistema deve soddisfare sia dal punto di vista software che dell'infrastruttura di rete.
Sulla base delle specifiche individuate, illustri quali possono essere le soluzioni possibili e scelga quella che a suo motivato giudizio è la piú idonea a rispondere alle specifiche indicate.
Il candidato, quindi, sviluppi l'intero progetto del sistema informatico, in particolare riportando: lo schema a blocchi dei moduli del prodotto software da realizzare; il progetto del database completo dello schema concettuale, dello schema logico, delle istruzioni DDL necessarie per l'implementazione fisica del database e di alcune query necessarie per sviluppare dei moduli software individuati; il codice di una parte significativa del software in un linguaggio di programmazione a scelta del candidato.

Dovendo gestire un’infrastruttura utilizzata anche da personale non esperto, bisogna per prima cosa assicurarsi che esso possa lavorare in sicurezza.

Ció significa porre i dipendenti di Media Way Spa in condizione di lavorare senza avere specifiche nozioni tecniche (se non quelle necessarie per l’utilizzo della piattaforma).

Il portale andrá utilizzato anche da persone esterno alla societá che quindi dovrá accedervi tramite un sistema di autenticazione raggiungibile mediante una connessione ad internet.

Davide Antonio Amodio – Classe VL – A.S. 2020/2021

Non potendo quindi limitare il raggiungimento del nostro servizio dovremo porre la massima attenzione nella gestione dei permessi e nel controllo degli accessi, sia all'infrastuttura che ai dati.

Difatti, per il servizio che offriremo dovremo conservare necessariamente diversi dati personali dei clienti e dei dipendenti che dovranno esser trattati nel rispetto del regolamento dell'UE 2016/679 e nel Dlgs 196/2003 oltre che conservare informazioni potenzialmente sensibili a livello aziendale.
Per quanto riguarda la protezione dell'infrastruttura utilizzeremo un firewall con una DMZ che ospiterá i server, con questo componente poniamo un primo filtro alle richieste in transito nella nostra rete.
L'accesso per la piattaforma amministrativa sará garantito unicamente in locale quindi per accedere dall'esterno sará necessario utilizzare la VPN aziendale.
La VPN, accessibile dall'esterno, sará configurata sulla porta 2000 utilizzando il protocollo UDP, tuttavia per ottenere la connessione sará necessario avere il certificato del server con le chiavi nel file dedicato (.ovpn) inserendo anche username e password personali.
Il firewall sará configurato per consentire quindi l'accesso alla VPN bloccando le altre connessioni in entrata.
Tramite le regole in uscita si impedirá anche agli impiegati di accedere ad altri siti malevoli.

Firewall main configuration					
Number	Protocol	Source IP	Destination IP	Destination Port	Action
1	ALL	0.0.0.0/0	0.0.0.0/0	0-65535	LOG
2	UDP	80.100.98.15/24	192.168.10.50/24	2000	DROP
3	UDP	0.0.0.0/0	192.168.10.50/24	2000	ALLOW
4	TCP	192.168.10.0/24	69.104.33.88/24	443	DROP
5	ALL	0.0.0.0/0	192.168.10.0/24	0-65535	DROP

All’arrivo di un pacchetto verranno applicate le regole nell’ordine descritto fin quando una di queste non risulterá compatibile con lo stesso.

Con la regola 5 chiudiamo tutte le connessioni in entrata,essa prende il nome di default policy e permette di gestire il pacchetto in tutti i casi non previsti dal nostro firewall.
La regola 4 invece é un blocco in uscita verso un IP malevolo.
La terza regola permette le connessioni in entrata verso la VPN mentre la seconda blocca la connessione in ingresso verso lo stesso IP, ció puó essere necessario se qualcuno sta tentando di forzare l’accesso.
La prima regola permette di generare un file di log in cui verranno conservate tutte le informazioni necessarie alla creazione di uno storico delle richieste in ingresso.

Va sottolineato che il blocco tramite IP quasi mai é la soluzione definitiva dato che molti host su internet non hanno un IP pubblico ma sono connessi sotto una rete con doppio NAT e che anche chi ha un host pubblico difficilmente lo conserva a lungo in quanto la maggior parte delle connessioni destinate ai privati non sono fornite di IP statico.

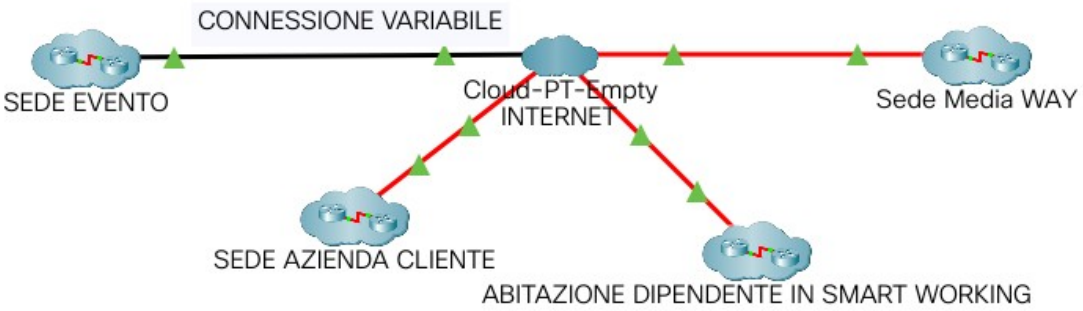
Nella DMZ sará consentito l’accesso al cluster sulle porte 80 e 443 TCP (per il server web) ma non al database o alle API che saranno accessibili solo in locale.

Firewall DMZ configuration					
Number	Protocol	Source IP	Destination IP	Destination Port	Action
1	ALL	0.0.0.0/0	0.0.0.0/0	0-65535	LOG
2	TCP	89.100.98.15/24	192.168.10.10/24	443	DROP
3	TCP	89.100.98.15/24	192.168.10.10/24	80	DROP
4	TCP	0.0.0.0/0	192.168.10.10/24	443	ALLOW
5	TCP	0.0.0.0/0	192.168.10.10/24	80	ALLOW
6	ALL	0.0.0.0/0	192.168.10.0/24	0-65535	DROP

Come nell'altra tabella la prima regola é di LOG e quindi permette il passaggio dei pacchetti salvandone l'header nello storico.
La seconda e la terza regola bloccano nello specifico un IP impedendo l'accesso alla piattaforma web mentre la quarta e la quinta permettono la fruizione del servizio.
L'ultima regola é la default policy per la DMZ che rifiuta tutti i pacchetti che non rientrano nelle regole precedenti.

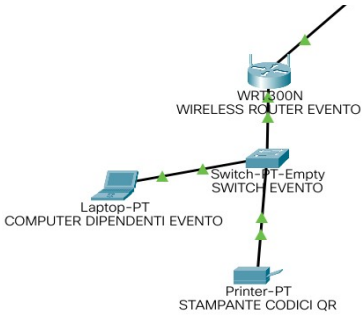
La VPN o Virtual Private Network permette, tramite la creazione di un tunnel criptato, una connessione remota con l'interno della rete aziendale.
Ció permette di evitare l'esposizione del pannello amministrativo ad internet.

Di seguito abbiamo gli screenshot della rete modellizzata in Cisco Packet Tracer:

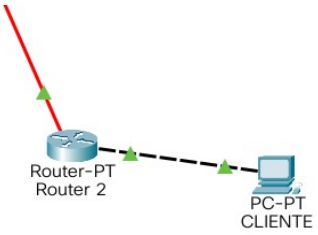


* la connessione con la sede dell'evento puó variare a seconda del luogo fisico nel quale l'evento si tiene

Postazione mobile dislocata nella sede fisica dell'evento:

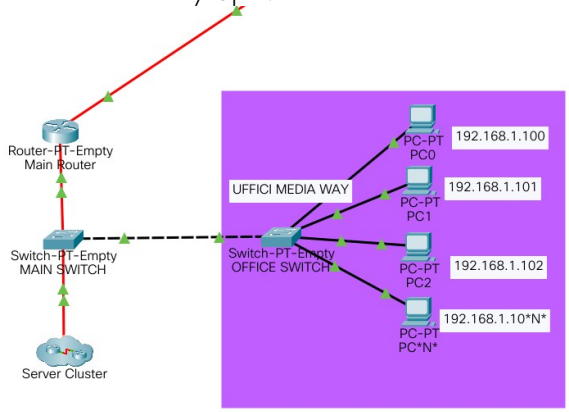


Postazione fissa connessa ad internet dalla quale il cliente puó accedere al sito:

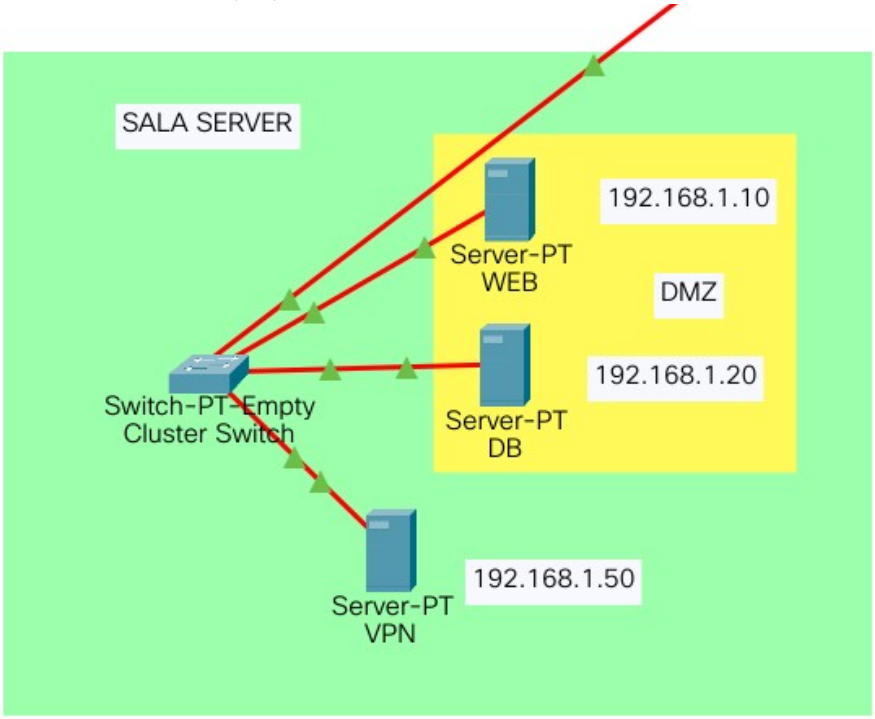


*la configurazione é uguale anche per il dipendente in smart working

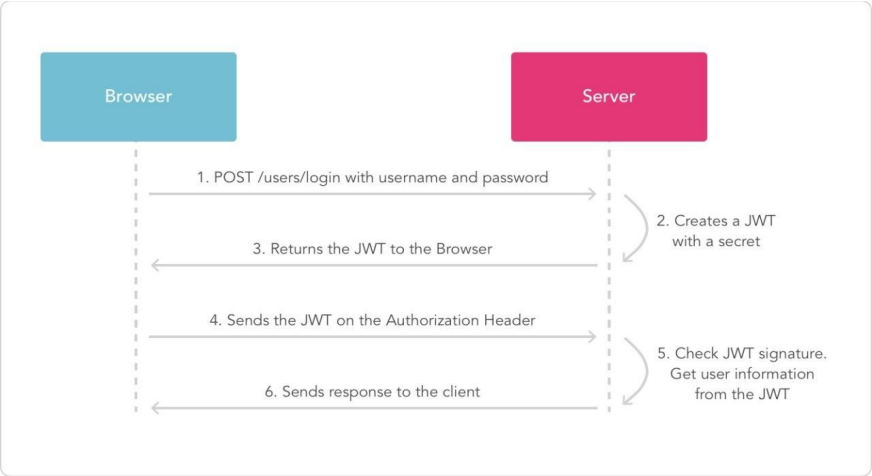
Modello degli uffici Media Way Spa:



Sala Server Media Way Spa:



Per assicurare la massima compatibilitá il sistema sará composto da un sito web per i clienti e da una WebApp per i dipendenti di Media Way Spa.
Come tecnologie e linguaggi saranno usati il PHP lato server, MySQL per le query sul database e python per le API locali necessarie al funzionamento della webApp.
Per il frontend, sia nella web app che sul sito saranno usati HTML5, CSS3 e JavaScript.
La webApp utilizzerá come framework React e funzionerá tramite chiamate alle API scritte in Python.
Per motivi di sicurezza dovremo verificare che tutte le richieste siano legittime e ció richiede un sistema di verifica ad ogni chiamata gestito dal server.
Il sistema di verifica scelto é JWT ovvero JSON Web Token ideale per lo sviluppo di web app basate su API RESTful.
Esso permette di verificare il gettone di accesso ad ogni chiamata, determinare la scadenza dello stesso ed eventualmente rinnovarlo.



Esempio di richiesta tramite JWT

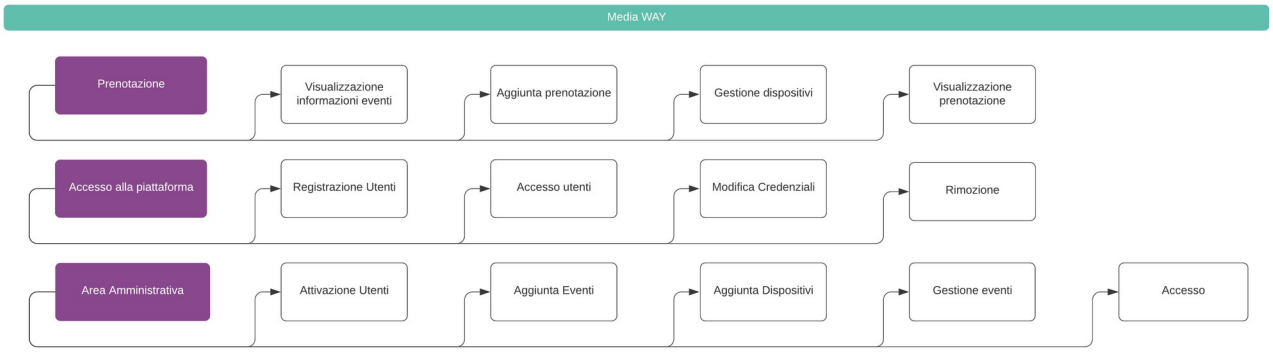
Il sito, utilizzando come web server Apache2, sará protetto anche tramite un web application firewall, ovvero mod_security che riconosce e blocca, analizzando le richieste, tentativi di SQL injection, XSS e altre richieste potenzialmente malevoli.

L'intero sistema é sviluppato utilizzando Docker che permette l'esecuzione del codice in vari ambienti isolati dalla macchina host. Con questa implementazione l'intero sistema viene eseguito in un ambiente isolato e facilmente trasportabile nel quale il software ha tutte le dipendenze necessarie.
Tramite Docker é anche possibile creare diverse interfaccie di rete virtuali gestendo con maggiore granularitá gli accessi.

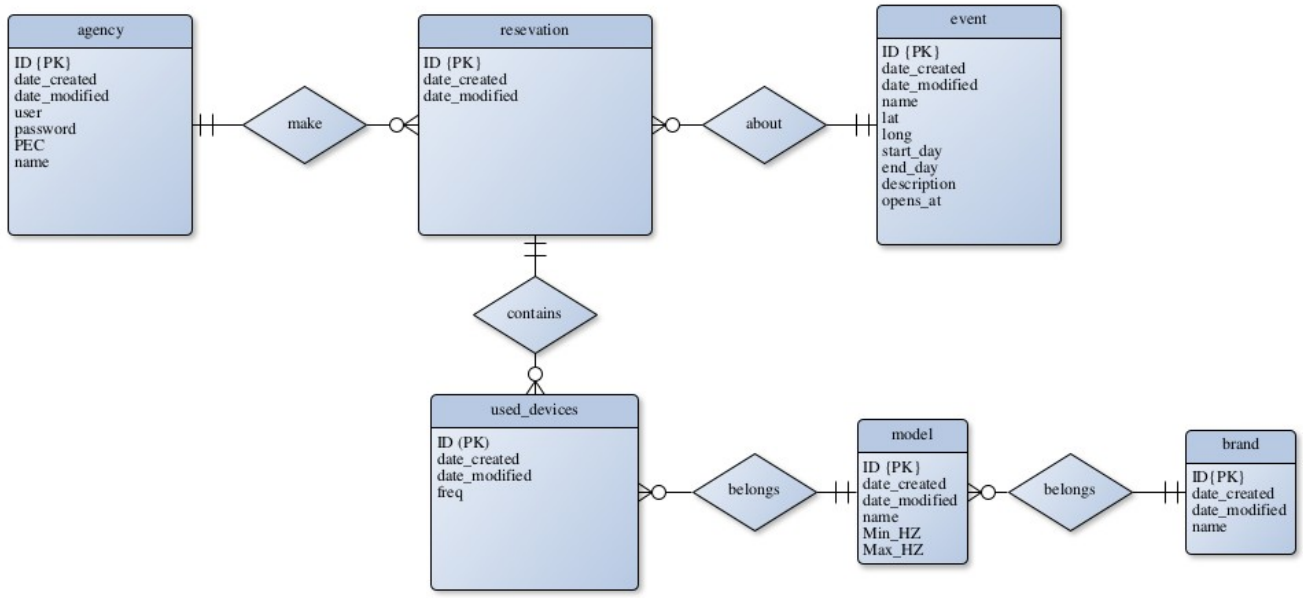
Il sistema presenta una classica architettura client-server ospitata fisicamente nella DMZ della rete con doppio backup del database, uno in locale e uno in remoto.
Il backup remoto andrá fatto ogni 24 ore in background senza quindi inficiare sul funzionamento del sistema mentre per quello in locale utilizzeremo dischi ZFS in RAID Z2 configurato come un unico VDEV nel cluster.

Per la registrazione sará necessario compilare l'apposito form sul sito inserendo una PEC come indirizzo di posta elettronica tuttavia prima di essere abilitati alla prenotazione delle frequenze negli eventi bisognerá essere accettati da un dipendente di Media Way Spa che potrá ricontattare il candidato per ulteriori informazioni.

Fatte le opportune ipotesi procedo con la creazione dello schema a blocchi:

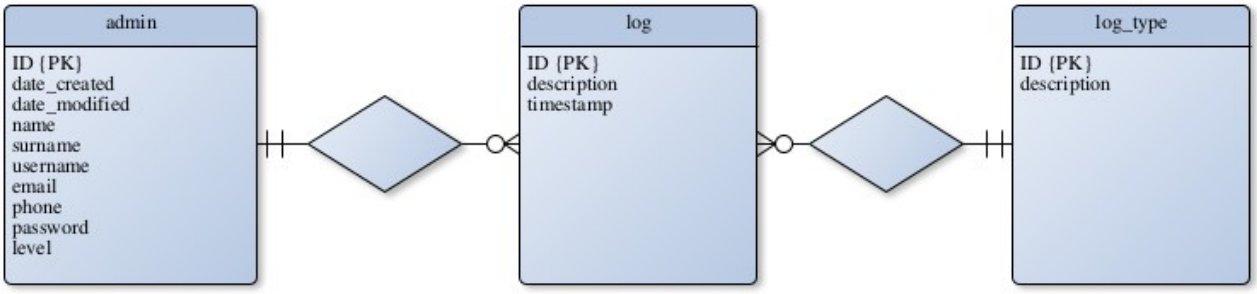


Lo schema ER del database é modellizzato nelle seguenti immagini:



Lo schema prevede che un'agenzia televisiva possa effettuare una prenotazione per ogni evento e che in ogni prenotazione si possano aggiungere uno o piú dispositivi.
Gli eventi sono creati dagli amministratori di Media Way Spa e hanno un timestamp per l'apertura delle iscrizioni.

Le entitá coinvolte nella gestione della piattaforma invece sono le seguenti:



Ogni amministratore puó generare dei log che devono rientrare in un tipo definito.

Con le opportune regole di derivazione possiamo generare lo schema logico:

AGENCIES

Nome	Tipo	Obbligatorio	Chiave	Default	Extra
ID	INT	SI	PK		A. I.
date_created	DATETIME	SI			
date_modified	DATETIME				
user	VARCHAR(64)	SI			
Password	VARCHAR(256)	SI			
PEC	VARCHAR(64)	SI			
name	VARCHAR(32)	SI			

EVENTS

Nome	Tipo	Obbligatorio	Chiave	Default	Extra
ID	INT	SI	PK		A. I.
date_created	DATETIME	SI			
date_modified	DATETIME				
name	VARCHAR(64)	SI			
lat	BIGINT	SI			
long	BIGINT	SI			
start_day	DATETIME	SI			
end_day	DATETIME	SI			
description	VARCHAR(4096)	SI			
opens_at	TIMESTAMP	SI			

RESERVATIONS

Nome	Tipo	Obbligatorio	Chiave	Default	Extra
ID	INT	SI	PK		A. I.
date_created	DATETIME	SI			
date_modified	DATETIME				
agency	INT	SI	FK		
event	INT	SI	FK		

BRANDS

Nome	Tipo	Obbligatorio	Chiave	Default	Extra
ID	INT	SI	PK		A. I.
date_created	DATETIME	SI			
date_modified	DATETIME				
name	VARCHAR(64)	SI			

MODELS

Nome	Tipo	Obbligatorio	Chiave	Default	Extra
ID	INT	SI	PK		A. I.
date_created	DATETIME	SI			
date_modified	DATETIME				
name	VARCHAR(64)	SI			
min_HZ	BIGINT	SI			
max_HZ	BIGINT	SI			

USED_DEVICES

Nome	Tipo	Obbligatorio	Chiave	Default	Extra
ID	INT	SI	PK		A. I.
date_created	DATETIME	SI			
date_modified	DATETIME				
freq	BIGINT	SI			
model	INT	SI	FK		
reservation	INT	SI	FK		

ADMIN

Nome	Tipo	Obbligatorio	Chiave	Default	Extra
ID	INT	SI	PK		A. I.
date_created	DATETIME	SI			
date_modified	DATETIME				
name	VARCHAR(32)	SI			
surname	VARCHAR(32)	SI			
username	VARCHAR(32)	SI			
email	VARCHAR(64)	SI			
phone	VARCHAR(16)	SI			
password	VARCHAR(256)	SI			
level	INT	SI			

LOG_TYPE

Nome	Tipo	Obbligatorio	Chiave	Default	Extra
ID	INT	SI	PK		A. I.
description	VARCHAR(1024)	SI			

LOG

Nome	Tipo	Obbligatorio	Chiave	Default	Extra
ID	INT	SI	PK		A. I.
description	VARCHAR(512)				
timestamp	TIMESTAMP				
log_type	INT	SI	FK		
admin	INT	SI	FK		

Per la creazione del database é stato implementato nel sistema di API uno script in python:

```
from app import db
from sqlalchemy.dialects.mysql import BIGINT



# Define a base model for other database tables to inherit
class Base(db.Model):
__abstract__ = True

id = db.Column(db.Integer, primary_key=True)
date_created = db.Column(db.DateTime, default=db.func.current_timestamp())
date_modified = db.Column(db.DateTime, default=db.func.current_timestamp(),
onupdate=db.func.current_timestamp())


class Admin(Base):
__tablename__ = 'admins'
```

```
name = db.Column(db.String(32), nullable=False)
surname = db.Column(db.String(32), nullable=False)
username = db.Column(db.String(64), nullable=False)
email = db.Column(db.String(256), nullable=False)
phone = db.Column(db.String(16), nullable=False)
password = db.Column(db.String(256), nullable=False)
level = db.Column(db.Integer, nullable=False, default=0)


class Log(Base):
__tablename__ = 'logs'

admin = db.Column(db.Integer, nullable=False)
description = db.Column(db.String(4096), nullable=False)
timestamp = db.Column(db.DateTime, nullable=False)
log_type = db.Column(db.Integer, nullable=False)



class LogType(Base):
__tablename__ = 'log_types'

description = db.Column(db.String(4096), nullable=False)


class Agency(Base):
__tablename__ = 'agencies'

username = db.Column(db.String(64), nullable=False)
password = db.Column(db.String(512), nullable=False)
PEC = db.Column(db.String(300), nullable=False)
name = db.Column(db.String(64), nullable=False)
enabled = db.Column(db.Boolean, nullable=False, default=0)

def toObject(self):
return {
'id': self.id,
'name': self.name,
'username': self.username,
'pec': self.PEC,
'is_active': self.enabled,
}


class Event(Base):
```

```
__tablename__ = 'events'

name = db.Column(db.String(128), nullable=False)
latitude = db.Column(db.Float, nullable=False)
longitude = db.Column(db.Float, nullable=False)
start_day = db.Column(db.DateTime, nullable=False)
end_day = db.Column(db.DateTime, nullable=False)
description = db.Column(db.String(4096), nullable=False)
opens_at = db.Column(db.DateTime, nullable=False)

def toObject(self):
return {
'id': self.id,
'name': self.name,
'description': self.description,
'location': [self.latitude, self.longitude],
'period': [self.start_day, self.end_day],
'opens_at': self.opens_at
}



class Reservation(Base):
__tablename__ = 'reservations'

agency = db.Column(db.Integer, nullable=False)
event = db.Column(db.Integer, nullable=False)

def toObject(self):
return {
'id': self.id,
'agency': Agency.query.get(self.agency).toObject(),
'event': Event.query.get(self.event).toObject(),
}



class UsedDevice(Base):
__tablename__ = 'used_devices'

freq = db.Column(BIGINT(unsigned=True), nullable=False)
reservation = db.Column(db.Integer, nullable=False)

def toObject(self):
return {
```

```
'id': self.id,
'freq': self.freq,
'reservation': Reservation.query.get(self.reservation).toObject(),
}


class Brand(Base):
__tablename__ = 'brands'

name = db.Column(db.String(64), nullable=False)

def toObject(self):
return {
'id': self.id,
'name': self.name,
}


class Model(Base):
__tablename__ = 'models'

name = db.Column(db.String(64), nullable=False)
brand = db.Column(db.Integer, nullable=False)
min_freq = db.Column(BIGINT(unsigned=True), nullable=False)
max_freq = db.Column(BIGINT(unsigned=True), nullable=False)

def toObject(self):
return {
'id': self.id,
'name': self.name,
'freq': [self.min_freq, self.max_freq],
'brand':Brand.query.get(self.brand).name,
}
```

La stessa operazione puó essere fatta tramite un file SQL:

```
CREATE DATABASE IF NOT EXISTS mediaway;
USE mediaway;

CREATE TABLE IF NOT EXISTS brands(
id INTEGER PRIMARY KEY AUTO_INCREMENT NOT NULL,
name VARCHAR(64) NOT NULL UNIQUE
);

CREATE TABLE IF NOT EXISTS models(
id INTEGER PRIMARY KEY AUTO_INCREMENT NOT NULL,
name VARCHAR(64) NOT NULL,
min_HZ BIGINT NOT NULL,
max_HZ BIGINT NOT NULL,
idBrand INTEGER NOT NULL,

FOREIGN KEY (idBrand) REFERENCES brands(id)
);

CREATE TABLE IF NOT EXISTS events(
id INTEGER PRIMARY KEY AUTO_INCREMENT NOT NULL,
name VARCHAR(128) NOT NULL,
latitude FLOAT NOT NULL,
longitude FLOAT NOT NULL,
start_day DATETIME NOT NULL,
end_day DATETIME NOT NULL,
description VARCHAR(4096) NOT NULL,
opens_at DATETIME NOT NULL
);

CREATE TABLE IF NOT EXISTS agencies(
id INTEGER PRIMARY KEY AUTO_INCREMENT NOT NULL,
username VARCHAR(64) NOT NULL UNIQUE,
password VARCHAR(512) NOT NULL,
pec VARCHAR(300) NOT NULL UNIQUE,
name VARCHAR(64) NOT NULL,
enabled BOOLEAN NOT NULL DEFAULT FALSE
);

CREATE TABLE IF NOT EXISTS reservations(
id INTEGER PRIMARY KEY AUTO_INCREMENT NOT NULL,
idEvent INTEGER NOT NULL,
idAgency INTEGER NOT NULL,

FOREIGN KEY (idEvent) REFERENCES events(id),
```

```
FOREIGN KEY (idAgency) REFERENCES agencies(id)
);

CREATE TABLE IF NOT EXISTS used_devices(
id INTEGER PRIMARY KEY AUTO_INCREMENT NOT NULL,
frequency BIGINT NOT NULL,
idReservation INTEGER NOT NULL,
idModel INTEGER NOT NULL,

FOREIGN KEY (idReservation) REFERENCES reservations(id),
FOREIGN KEY (idModel) REFERENCES models(id),
);
```

Le operazioni fondamentali di seguito inserite sono la connessione al database, il form di login, il form per la registrazione, la classe per l'oggetto di tipo Agency e la classe DAO per interfacciarla al DBMS:

connection.php

```php
<?php
class Connection {
const HOSTNAME = '172.16.4.20';
const DB = 'mediaway';
const USER = 'user'; #change in production
const PASSWORD = 'verydifficultpassword'; #change in production
private static $conn = null;
public static function getConnection() {
$dsn = "mysql:host=" . self::HOSTNAME . ";dbname=" . self::DB;
if (self::$conn == null) {
try {
self::$conn = new PDO($dsn, self::USER, self::PASSWORD);
self::$conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
if(false){
echo "Successfully connected:<br>" . PHP_EOL;
}
} catch (PDOException $e) {
throw $e;
}
}
return self::$conn;
}
}
```

login.php

```php
<?php
session_start();
if (isset($_SESSION['user_id'], $_SESSION['username'])) :
header("Location: ../index.php");
endif;

if (isset($_POST['login_btn'])) :

if (empty($_POST['username']) || empty($_POST['password'])) :
$_SESSION['msg_txt'] = "Inserisci username e password.";
$_SESSION['msg_type'] = "error";
else :
$username = trim($_POST['username']);
$password = trim($_POST['password']);
try {
include_once __DIR__ . '/model/DAO/classes/connection.php';
$sql = "
SELECT *
FROM agencies
WHERE username = :username";
$conn = Connection::getConnection();

$stm = $conn->prepare($sql);
$stm->bindParam(':username', $username, PDO::PARAM_STR);
$stm->execute();
$record = $stm->fetch(PDO::FETCH_ASSOC);
if (!$record || password_verify($password, trim($record['password'])) === false) {
$_SESSION['msg_txt'] = "Credenziali utente errate.";
$_SESSION['msg_type'] = "error";
$logged = true;
} else {
$_SESSION['user_id'] = $record['id'];
$_SESSION['username'] = $username;
header('Location: ../index.php');
exit;
}
} catch (PDOException $e) {
$_SESSION['msg_txt'] = 'Errore sul server: ' . $e->getMessage();
$_SESSION['msg_type'] = 'error';
}
endif;
endif;
?>
```

```
<!DOCTYPE html>
<html lang="it">
<head>
<meta charset="UTF-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<link href="https://unpkg.com/tailwindcss@^2/dist/tailwind.min.css" rel="stylesheet">
<link href="<?php __DIR__ ?> /res/css/style.css" rel="stylesheet">

<title>Media Way Spa - login</title>
</head>
<body class="">
<?php include_once __DIR__ . '/components/navbar.php' ?>

<link
rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.13.0/css/all.min.css"
/>
<div class="md:bg-gradient-to-r bg-gradient-to-b from-green-400 to-blue-100 rounded-2xl
mb-10 flex mx-8 lg:mx-32 md:mx-80">
<div class="flex-col flex ml-auto mr-auto items-center w-full lg:w-2/3 md:w-3/5">
<h1 class="font-bold text-2xl my-10 text-white"> Login </h1>
<form action="<?php $_SERVER['PHP_SELF']?>" method="post" class="mt-2 flex flex-col
lg:w-1/2 w-8/12">
<div class="flex flex-wrap items-stretch w-full mb-4 relative h-15 bg-white items-center
rounded mb-6 pr-10">
<div class="flex -mr-px justify-center w-15 p-4">
<span
class="flex items-center leading-normal bg-white px-3 border-0 rounded rounded-r-none text-
2xl text-gray-600"
>
<i class="fas fa-user-circle"></i>
</span>
</div>
<input
type="text"
class="flex-shrink flex-grow flex-auto leading-normal w-px flex-1 border-0 h-10 border-grey-
light rounded rounded-l-none px-3 self-center relative font-roboto text-xl outline-none"
placeholder="username"
name="username"
/>
</div>
<div class="flex flex-wrap items-stretch w-full relative h-15 bg-white items-center rounded
mb-4">
<div class="flex -mr-px justify-center w-15 p-4">
```

```
<span
class="flex items-center leading-normal bg-white rounded rounded-r-none text-xl px-3
whitespace-no-wrap text-gray-600"
>
<i class="fas fa-lock"></i>
</span
>
</div>
<input
id="passContainer"
type="password"
class="flex-shrink flex-grow flex-auto leading-normal w-px flex-1 border-0 h-10 px-3 relative
self-center font-roboto text-xl outline-none"
placeholder="password"
name="password"
/>
<div class="flex -mr-px">
<span
class="flex items-center leading-normal bg-white rounded rounded-l-none border-0 px-3
whitespace-no-wrap text-gray-600"
>
<i onclick="showPassword()" id="passToggle" class="fas fa-eye-slash"></i>
</span>
</div>
</div>
<a href="#" class="text-base text-white text-right font-roboto leading-normal
hover:underline mb-6">Password dimenticata?</a>
<input
name="login_btn"
type="submit"
value="Login"
class="bg-purple-600 font-semibold uppercase py-4 text-center px-17 md:px-12 md:py-4 text-
white rounded-lg leading-tight text-xl md:text-base font-sans mt-4 mb-20"
>
</input>
</form>
</div>
</div>

<?php include_once __DIR__ . '/components/footer.php' ?>
<script src="<?php __DIR__?> /res/js/functions.js"></script>
</body>
</html>
```

signup.php

```php
<?php
session_start();
include_once __DIR__ . '/model/DAO/classes/connection.php';
if(isset($_POST['submit'])){
if(empty($_POST['name']) || empty($_POST['username']) || empty($_POST['pec']) ||
empty($_POST['password'])){
$_SESSION['msg_txt'] = "Almeno un campo non é stato compilato correttamente.";
$_SESSION['msg_type'] = "error";
}
else{
$username = trim($_POST['username']);
$password = trim($_POST['password']);
$pec = trim($_POST['pec']);
$validUsername = filter_var($username,FILTER_VALIDATE_REGEXP,["options" => ["regexp"
=> "/^[a-z\d_.]{4,20}$/i"]]);
$pwdLenght=strlen($password);
if(!$validUsername){
$_SESSION['msg_txt'] = "Username non valida. Sono ammessi solo caratteri
alfanumerici, l'underscore e il punto. Lunghezza minima 5 caratteri.
Lunghezza massima 20 caratteri";
$_SESSION['msg_type'] = "warning";
}
elseif ($pwdLenght<3 && $pwdLenght > 20) {
$_SESSION['msg_txt'] = "Password non valida. Lunghezza minima 8 caratteri.
Lunghezza massima 20 caratteri";
$_SESSION['msg_type'] = "warning";
}
else{
try {
$now = date("y-m-d h:i:s");
include_once __DIR__ . '/model/DAO/classes/connection.php';
$password_hash = password_hash($password, PASSWORD_DEFAULT);

$query = "
INSERT INTO agencies (date_created, username, password, PEC, name)
VALUES (:date_created, :username, :password, :PEC, :name)";
$conn = Connection::getConnection();
$stm = $conn->prepare($query);
$stm->bindParam(':date_created',$now, PDO::PARAM_STR);
$stm->bindParam(':username', $username, PDO::PARAM_STR);
$stm->bindParam(':password', $password_hash, PDO::PARAM_STR);
$stm->bindParam(':PEC', $_POST['pec'], PDO::PARAM_STR);
$stm->bindParam(':name', $_POST['name'], PDO::PARAM_STR);
$stm->execute();
```

```
header("Location: index.php");

if ($stm->rowCount() > 0) :
$_SESSION['msg_txt'] = "Registrazione eseguita con successo";
$_SESSION['msg_type'] = "success";
else :
$_SESSION['msg_txt'] = "Si é verificato un problema con l'inserimento dei dati";
$_SESSION['msg_type'] = "error";
endif;
} catch (PDOException $e) {
$_SESSION['msg_txt'] = "Attenzione, l'utente non è stato registrato: " . $e->getMessage();
$_SESSION['msg_type'] = 'error';
}
}
}
}
?>

<!DOCTYPE html>
<html lang="it">
<head>
<meta charset="UTF-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<link href="https://unpkg.com/tailwindcss@^2/dist/tailwind.min.css" rel="stylesheet">
<link href="<?php __DIR__ ?> /res/css/style.css" rel="stylesheet">

<title>Media Way Spa - registrati</title>
</head>
<body class="">
<?php include_once __DIR__ . '/components/navbar.php' ?>

<link
rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.13.0/css/all.min.css"
/>
<div class="md:bg-gradient-to-r bg-gradient-to-b from-green-400 to-blue-100 rounded-2xl
mb-10 flex mx-16 lg:mx-64 md:mx-32">
<div class="flex-col flex ml-auto mr-auto items-center w-full lg:w-2/3 md:w-3/5">
<h1 class="font-bold text-2xl my-10 text-white"> Registrati </h1>
<form action="<?php $_SERVER['PHP_SELF']?>" method="post" class="mt-2 flex flex-col
lg:w-3/4 w-10/12">
<div class="flex flex-wrap items-stretch w-full mb-4 relative h-15 bg-white items-center
rounded mb-6 pr-10">
<div class="flex -mr-px justify-center w-15 p-4">
```

```
<span
class="flex items-center leading-normal bg-white px-3 border-0 rounded rounded-r-none text-2xl text-gray-600"
>
<i class="fas fa-building"></i>
</span>
</div>
<input
type="text"
name="name"
class="flex-shrink flex-grow flex-auto leading-normal w-px flex-1 border-0 h-10 border-grey-light rounded rounded-l-none px-3 self-center relative font-roboto text-xl outline-none"
placeholder="nome azienda"
/>
</div>
<div class="flex flex-wrap items-stretch w-full mb-4 relative h-15 bg-white items-center rounded mb-6 pr-10">
<div class="flex -mr-px justify-center w-15 p-4">
<span
class="flex items-center leading-normal bg-white px-3 border-0 rounded rounded-r-none text-2xl text-gray-600"
>
<i class="fas fa-user-circle"></i>
</span>
</div>
<input
type="text"
class="flex-shrink flex-grow flex-auto leading-normal w-px flex-1 border-0 h-10 border-grey-light rounded rounded-l-none px-3 self-center relative font-roboto text-xl outline-none"
placeholder="username"
name="username"
/>
</div>
<div class="flex flex-wrap items-stretch w-full mb-4 relative h-15 bg-white items-center rounded mb-6 pr-10">
<div class="flex -mr-px justify-center w-15 p-4">
<span
class="flex items-center leading-normal bg-white px-3 border-0 rounded rounded-r-none text-2xl text-gray-600"
>
<i class="far fa-envelope"></i>
</span>
</div>
<input
type="text"
```

```
class="flex-shrink flex-grow flex-auto leading-normal w-px flex-1 border-0 h-10 border-grey-light rounded rounded-l-none px-3 self-center relative font-roboto text-xl outline-none"
placeholder="PEC"
name="pec"
/>
</div>
<div class="flex flex-wrap items-stretch w-full relative h-15 bg-white items-center rounded mb-4">
<div class="flex -mr-px justify-center w-15 p-4">
<span
class="flex items-center leading-normal bg-white rounded rounded-r-none text-xl px-3 whitespace-no-wrap text-gray-600"
>
<i class="fas fa-lock"></i>
</span
>
</div>
<input
id="passContainer"
type="password"
class="flex-shrink flex-grow flex-auto leading-normal w-px flex-1 border-0 h-10 px-3 relative self-center font-roboto text-xl outline-none"
placeholder="password"
name="password"
/>
<div class="flex -mr-px">
<span
class="flex items-center leading-normal bg-white rounded rounded-l-none border-0 px-3 whitespace-no-wrap text-gray-600"
>
<i onclick="showPassword()" id="passToggle" class="fas fa-eye-slash"></i>
</span>
</div>
</div>
<input
type="submit"
name="submit"
value="Registrati"
class="bg-purple-600 font-semibold uppercase py-4 text-center px-17 md:px-12 md:py-4 text-white rounded-lg leading-tight text-xl md:text-base font-sans mt-4 mb-20"
>
</input>
</form>
<?php
if(isset($_SESSION['msg_type'])){
```

```
echo($_SESSION['msg_txt']);
}
?>
</div>

</div>

<?php include_once __DIR__ . '/components/footer.php' ?>
<script src="<?php __DIR__?> /res/js/functions.js"></script>
</body>
</html>
```

agency.php

```
<?php

class Agency{
private int $id;
private string $username;
private string $password;
private string $pec;
private string $name;
private bool $enabled;

function __construct(int $id, string $username,string $password,string $pec,string $name,bool
$enabled){
$this->id = $id;
$this->username = $username;
$this->password = password_hash($password, PASSWORD_DEFAULT);
$this->pec = $pec;
$this->name = $name;
$this->enabled = $enabled;
}

function getId(){
return $this->id;
}
function getUsername(){
return $this->username;
}
function getPassword(){
return $this->password;
}
function getPec(){
return $this->pec;
```

```
}
function getName(){
return $this->name;
}
function isEnabled(){
return $this->enabled;
}

function setId(int $id){
$this->id = $id;
}
function setUsername(string $username){
$this->username = $username;
}
function setPassword(string $password){
$this->password = password_hash($password, PASSWORD_DEFAULT);
}
function setPec(string $pec){
$this->pec = $pec;
}
function setName(string $name){
$this->name = $name;
}
}

?>
```

agencyDAO.php

```
<?php
include_once __DIR__ . '/classes/connection.php';
include_once __DIR__ . '/classes/agency.php';

class agencyDAO{
static function getAgency(int $id): Agency{
$sql = "SELECT name, PEC, enabled"
. " FROM agencies WHERE id = NULLIF(:id, '')";

try {
$conn = Connection::getConnection();
$stm = $conn->prepare($sql);
$stm->bindParam(':id', $id, PDO::PARAM_INT);
$stm->execute();
if (!$row = $stm->fetch(PDO::FETCH_ASSOC)) {
throw new PDOException('<strong>agenzia non trovata</strong>');
}
```

```
$name = $row['name'];
$pec = $row['PEC'];
$enabled = $row['enabled'];

$agency = new Agency($id, $_SESSION['username'], '', $pec, $name,$enabled);
} catch (PDOException $e) {
throw $e;
} catch (Exception $e) {
throw new Exception("<strong>errore imprevisto,</strong>");
}
return $agency;
}

static function isInEvent(int $eventId,int $agencyId):bool{
$sql = "SELECT $agencyId IN (SELECT agency FROM reservations WHERE event = $eventId)
AS res";
try {
$conn = Connection::getConnection();
$stm = $conn->prepare($sql);
$stm->execute();
if (!$row = $stm->fetch(PDO::FETCH_ASSOC)) {
throw new PDOException('<strong>errore</strong>');
}
return $row['res'] != 0;
}
catch (Exception $e) {
throw new Exception("<strong>errore imprevisto</strong>");
}
}
}
?>
```

Per simulare il cluster ed anche per una maggior velocitá di sviluppo il file che descrive i container docker presenta anche i domini e le reti (virtuali) con i rispettivi accessi garantiti seguendo l'analisi sviluppata all'inizio:

docker-compose.yaml

```
version: '3.4'

services:
api:
hostname: 'api.mediaway.com'
build: './api/'
links:
- 'database'
volumes:
- './api/code:/usr/src/app:z'
networks:
api_net:
ipv4_address: 172.16.1.5
database_net:
ipv4_address: 172.16.4.80
# admin:
# hostname: 'admin.mediaway.com'
# build: './admin/'
# volumes:
# - './admin/code:/usr/src/app:ro'
#
# links:
# - 'api'
#
# networks:
# administrative_net:
# ipv4_address: 172.16.2.5

www:
hostname: 'mediaway.com'
build: './www/'
volumes:
- './www/code:/var/www/html:z'
links:
- 'database'
networks:
www_net:
ipv4_address: 172.16.3.10 #192.168.1.10
database_net:
ipv4_address: 172.16.4.70
pma:
hostname: 'pma.mediaway.com'
image: phpmyadmin
environment:
```

```
PMA_ARBITRARY: 1
PMA_HOST: database
links:
- 'database'
ports:
- '443:443'
- '80:80'
networks:
database_net:
ipv4_address: 172.16.4.69 #192.168.1.20
database:
image: 'mysql:latest'
command: --default-authentication-plugin=mysql_native_password
environment:
MYSQL_ROOT_PASSWORD: rootPWD #change in production
MYSQL_DATABASE: mediaway
MYSQL_USER: user #change in production
MYSQL_PASSWORD: verydifficultpassword #change in production

volumes:
- sql_database:/var/lib/mysql:rw
networks:
database_net:
ipv4_address: 172.16.4.20 #192.168.1.20

volumes:
sql_database:

networks:
www_net:
ipam:
driver: default
config:
- subnet: 172.16.3.0/24
api_net:
ipam:
driver: default
config:
- subnet: 172.16.1.0/24

database_net:
ipam:
driver: default
config:
- subnet: 172.16.4.0/24
```

```
administrative_net:
ipam:
driver: default
config:
- subnet: 172.16.2.0/24
```

Dovendo lavorare sul progetto ho utilizzato il sistema Git sulla piattaforma GitHub, il software permette di sviluppare in gruppo fornendo anche la possibilitá di assegnare e dividere i compiti, di impostare milestone permettondo quindi la gestione e lo sviluppo del progetto anche su piú dispositivi e fra piú sviluppatori.
Di seguito una rappresentazione del mio workflow con tutte le modifiche caricate:

```
* 69ce38f (HEAD -> main, origin/main, origin/HEAD) elaborato.odt
* aebf23a added stuff on relazione.odt
* c086d7e relazione.odt
* da462f3 added remove device functionality, added automatic reservation remover if the customer removes all the devices, added all the needed DAO functions
* c1da2da added the add device functionality for the already booked accounts, started the remove device implementation (needs the query on the DB and the function on the DAOclass) still missing the book functionality
* 55c4a9c added remaining dao classes and methods, still missing add device or remove
* 84c953f added create event functionality, failed to add a map, create the manage event page (all on react app), added flask APIs for this functionalities
* fec3310 implemented more APIs, implemented add device, add brand, enabling user started add event implementation (broke the whole react app, fixed the whole react app)
* eebfc4e implemented more API endpoints, added stuff on the react app
* 2e6bdb3 working on reservation.php, fixed serous issues
* 0c7a729 added modelDAO, bug fixes around the model class, still developing reservation.php
* a9524e5 added agencyDAO, eventDAO, several critical bugfixes, working on reservation.php (not finished)
* 1016514 renamed website to cluster since the docker part now is the whole server, minor bug fix, added about-us.php
* a62a787 fixed some serious session issues, started signup.php (not working but i still don't know why)
* abcc754 added login, minor html and php adjustements
* b98edfc started API implementation with the react app created agency class and event class (not tested)
* ba14990 added login page, footer minor fix
* d5adcfd footer fix
* 7e644df working on frontend website
* 1bdc172 started website
* b5f5e57 fixed docker bugs (website was not able to access to the DB), started website developing
*   d1cf20e Merge branch 'main' of https://github.com/WAPEETY/media-way
|\
| * e33a5ad Delete www directory
| * 2a8647b Delete docker-compose.yaml
* | ad9c5f4 minor fix on SQL file DBcreate added APIs (not tested)
|/
* 65675ae refactored docker compose, and all the docker system
* bc15def added calendar, modified cards minor bug fixes on the media-admin react app
* 1c91cae addedd cards, need to know how grids works
* 94c72e2 started the frontend developing for the react app
* 9049f8a minor changes
* c0e6aa4 started react app
* ad025d9 started react app
* 293c7d1 finshed the network (refactored network/rete.pkt, included screenshots in the sub folder), modified relazione.odt, moved ER screen and files to the model folder
* aebd5c9 finished sistemi and TPSIT, working on ER, missing network scheme
* 15630bf added assignment, style adjustments
* df85593 ended firewall started VPN
* 5c2a8c5 added GDPR stuff on relazione.odt
* a1327b7 firewall config in relazione.odt
* 14a9954 changed stuff in relazione, made some stuff on ER relationships
* f9b81f3 still working on relazione
* cffec19 ZFS in relazione
* 3fdbf3e added stuff on the final document
* 5c42da0 made some changes at the ER scheme
* 85d43d5 basic docker container and first demo of the ER scheme in the proof_of_concept folder
:
```