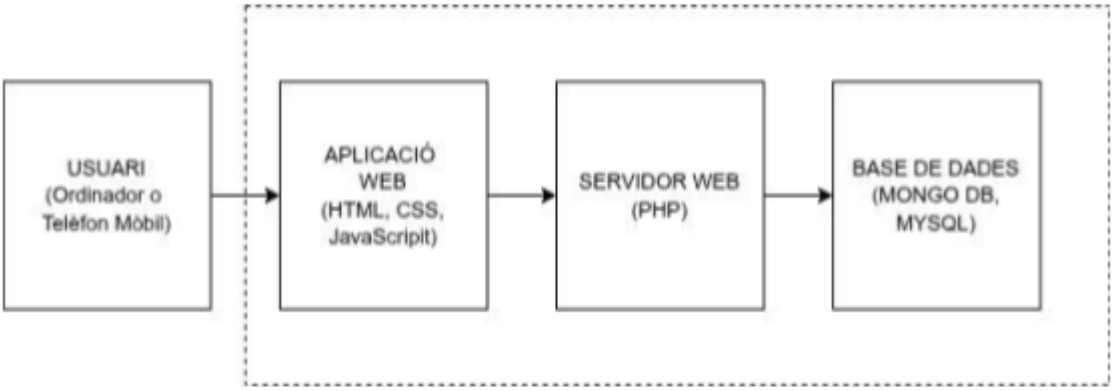
SISTEMES INFORMÀTICS

- Arquitectura del sistema:



1. Gestió d'Incidències

- Aquest projecte és una aplicació web per gestionar incidències mitjançant tècnics que resolen les incidències i administradors que asignen i gestionen les incidències. L'objectiu principal és poder fer que la comunicació entre els membres de l'organització sigui clara i ordenada..

2. Tipus d'Usuari

- L'aplicació té diferents rols amb accesos a diiferents llocs de l'aplicació (encara que no hem pogut fer el login per poder implementar-ho de forma funcional).

  Els diferents rols són :

  - Usuaris : Les seves funcions són crear incidències i veure l'estat d'una incidència.
  - Tècnics : Les seves funcions són fer actuacions en les incidències que tenen assignades i resoldre-les.
  - Administradors : Les seves funcions són assignar les incidències a un tècnic, veure les actuacions dels tècnics i gestionar si s'han de tancar les incidències.

SISTEMES INFORMÀTICS

3. Documentació docker-compose.yaml

- Aquest fitxer docker-compose.yaml defineix i gestiona diversos contenidors Docker per desplegar el sistema de gestió d'incidències.

- Els serveis principals són:

  - Aplicació Web amb PHP i Apache (apache_php_mysql_mongodb) : Aquest contenidor executa l'aplicació web amb PHP i Apache. Carrega automàticament les dependències de composer. Exposa el port 8080, accessible des del navegador. Munta el directori php/ a /var/www/html, permetent que el codi PHP es desplegui dins del contenidor.

```yaml
services:
  apache_php_mysql_mongodb:
    #La imatge php:apache no té tots els drivers. hem de fer una imatge a mida
    # Podeu fer servir el dockerfile que hi ha a /images i fer el build
    # o podeu agafar aquest imatge, ebota/daw:apache_php_mysql_mongodb,
    #  que ja està construüida i està a dockerhub. Teniu més info a:
    # https://hub.docker.com/r/ebota/daw
    image: ebota/daw:apache_php_mysql_mongodb
    ports:
      - "8080:80"
    volumes:
      - ./php:/var/www/html
    # És necessari que hi hagi el fitxer composer.json a la carpeta php i la següent directiva
    # perque el php instal·li les dependències de mongo
    command: /bin/bash -c "composer install --no-interaction --no-plugins --no-scripts && echo WEBSERVER_____UP_____AND__
    environment:
      VAR1: "sóc una pera i estic al docker-compose"
      VAR2: ${VAR2}
    depends_on:
      - db
```

  - Base de dades MySQL (db) : Contenidor que executa MySQL per emmagatzemar incidències, tècnics i usuaris. Les credencials estan definides a environment. Les dades de la BD es guarden en volums persistents (db_data) per evitar pèrdues. No exposa ports externs per seguretat, ja que l'aplicació web accedeix internament.

```yaml
db:
  # Tota la informació d'aquesta imatge està https://dockerhub.com/_/mysql
  image: mysql:9.3
  environment:
    MYSQL_ROOT_PASSWORD: passwordDeRootQueNoShaDeFerServirMai
    # És millor no crear la BBDD aquí, ja que no pots control·lar la codificació
    # de caràcters i aleshores donarà problemes amb accents i caràcters especials
    # La BBDD es crearà a l'inici del contenidor amb els script d'inicialització
    # MYSQL_DATABASE: a24biedommar_ProjecteFinal_MySql
    MYSQL_USER: usuari
    MYSQL_PASSWORD: paraula_de_pas
    LANG: C.UTF-8
  #    El mysql no s'exposa a l'exterior
  #    L'aplicació web hi accedirà per la xarxa interna de docker
  #    ports:
  #      - "3306:3306"

  # La carpeta de mysql ha d'estar al .gitignore per no pujar-la al repositori
  volumes:
    - ./db_data:/var/lib/mysql
    - ./db_init:/docker-entrypoint-initdb.d
```

- Adminer (Interfície web per gestionar MySQL) (adminer) :
Aquest contenidor és una interfície web per accedir i administrar la base de dades MySQL. Accessible via http://localhost:8081. Utilitza les mateixes credencials de MySQL per iniciar sessió.

```yaml
adminer:
  image: adminer
  ports:
    - "8081:8080"
  depends_on:
    - db
```

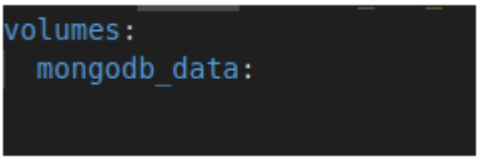
- Base de dades NoSQL MongoDB (mongo): Contenidor que executa MongoDB per gestionar dades no relacionals. Exposa el port 27017, utilitzat per connexions internes. Guarda les dades en un volum persistent (mongodb_data) per evitar pèrdues.

```yaml
mongo:
  image: mongo:latest
  volumes:
    #Named volume, no cal posar res al .gitignore
    #perque no es crea una carpeta al host sinó que es crea un volum a docker
    # i es pot veure amb docker volume ls
    # i es pot eliminar amb docker volume rm nom_del_volum
    - mongodb_data:/data/db
  ports:
    - "27017:27017"
  environment:
    MONGO_INITDB_ROOT_USERNAME: root
    MONGO_INITDB_ROOT_PASSWORD: example
```

- Interfície de gestió Mongo Express (mongo-express) :Aquest contenidor proporciona una interfície web per gestionar MongoDB. Accessible via http://localhost:8082. Permet veure i administrar les col·leccions de la BD NoSQL.

```yaml
mongo-express:
  image: mongo-express
  ports:
    - "8082:8081"
  environment:
    #Per entrar a la web de mongo-express, les credencials són
    #admin
    #pass
    ME_CONFIG_MONGODB_URL: mongodb://root:example@mongo:27017/
  depends_on:
    - mongo
```

Per evitar pèrdues de dades quan es reinicien els contenidors, les bases de dades usen volums Docker. MySQL → db_data MongoDB → mongodb_data



4. Base de Dades

El sistema utilitza una base de dades relacional (MySQL) per gestionar les incidències i els usuaris.

- La base de dades conté les següents taules:
  - Usuari → Registra els usuaris que poden crear incidències.
  - Incidència → Guarda la informació dels problemes reportats.
  - Prioritat → Defineix la urgència de cada incidència (Baixa, Mitjana, Alta).
  - Tipus→ Indica el tipus d'incidència.
  - Estat → Indica si una incidència està pendent, en procés o resolta.
  - Tècnic → Assigna un professional per resoldre incidències.
  - Departament → Cada incidència pertany a un departament específic.
  - Actuació → Registra les actucions fetes pels tècnics per solucionar una incidència.

5. Manteniment del Sistema

   Per garantir el bon funcionament de l'aplicació, cal optimitzar la gestió de recursos i prevenir errors.

   Optimització del sistema :

   - Monitorització dels contenidors Docker → Revisar l'estat amb docker ps.
   - Gestió de logs → Analitzar errors amb docker logs (i el nom).

   Resolució de problemes comuns :

   - Si l'aplicació no respon, aleshores s'ha de comprovar que docker-compose up ha iniciat correctament. I Revisar els logs per detectar possibles errors (docker logs apache_php_mysql_mongodb).
   - Si hi ha error en la connexió MySQL es verifica si el contenidor db està en marxa (docker ps). I confirma que les credencials de connexio.php són correctes.

6. Còpies de Seguretat amb Docker

   Les còpies de seguretat són una mesura essencial per garantir la protecció i recuperació de dades en cas d'errors o pèrdues inesperades. En el sistema de gestió d'incidències, és fonamental disposar d'un mecanisme per guardar regularment la informació de la base de dades.

   Les còpies de seguretat permeten restaurar la base de dades a un estat anterior en cas de fallades del sistema, errors humans o problemes amb el servidor. Aquest procés és crucial per evitar la pèrdua d'informació sobre incidències, tècnics, usuaris i altres elements del sistema.

   Per garantir la seguretat de les dades, es recomana seguir un esquema de backup automatitzat. Això inclou:

   - Còpies diàries per assegurar que sempre es pot restaurar una versió recent de la base de dades.
   - Còpies externes en un servidor segur per evitar pèrdua de dades en cas de problemes amb el sistema principal.

Manteniment dels backups en diversos formats per garantir compatibilitat i facilitat de restauració.

En cas d'una fallada del sistema, es pot restaurar la base de dades mitjançant una còpia de seguretat recent. Aquest procés permet recuperar totes les incidències i la informació associada, evitant la interrupció del servei per als usuaris i tècnics.