B

ADELAIDE DONOVAN
CRETE JONATHAN

GITHUB

https://github.com/adelaidedonovan

https://github.com/CrtJonathan

MISE EN SITUATION PROFESSIONNEL

PRESENTATION

L'ENTREPRISE DUNORD DISPOSE D'UNE QUINZAINE D'EMPLOYES PRODUISANT DES JOUETS.

ELLE EST COMPOSEE DE 3 SERVICES PRINCIPAUX : LE SERVICE ADMINISTRATIF, DE PRODUCTION, AINSI QUE LE SERVICE COMMERCIAL.

CONTEXTE INITAL

M.EDGAR DUNORD A DECIDE DE DEVELOPPER SON ENTREPRISE EN CREANT UN SITE VITRINE, UN AUTRE D'E-COMMERCE… AUSSI IL VOUS A EMBAUCHE COMME PRINCIPAL ET UNIQUE ACTEUR DU SERVICE INFORMATIQUE.

LA MISSION

M.EDGAR DUNORD VOUS DEMANDE D'AMELIORER LES OUTILS PERMETTANT D'ASSURER LA MAINTENANCE MATERIELLE ET LOGICIELLE DU PARC.

Formalité de livraison

Rapport envoyé par mail : alain.dehors@ac-creteil.fr
Format du rapport : Dunord-nom(s)-auteur(s).zip (pas plus de 2
Par groupe, avec explications sur la répartition du travail réalisé)

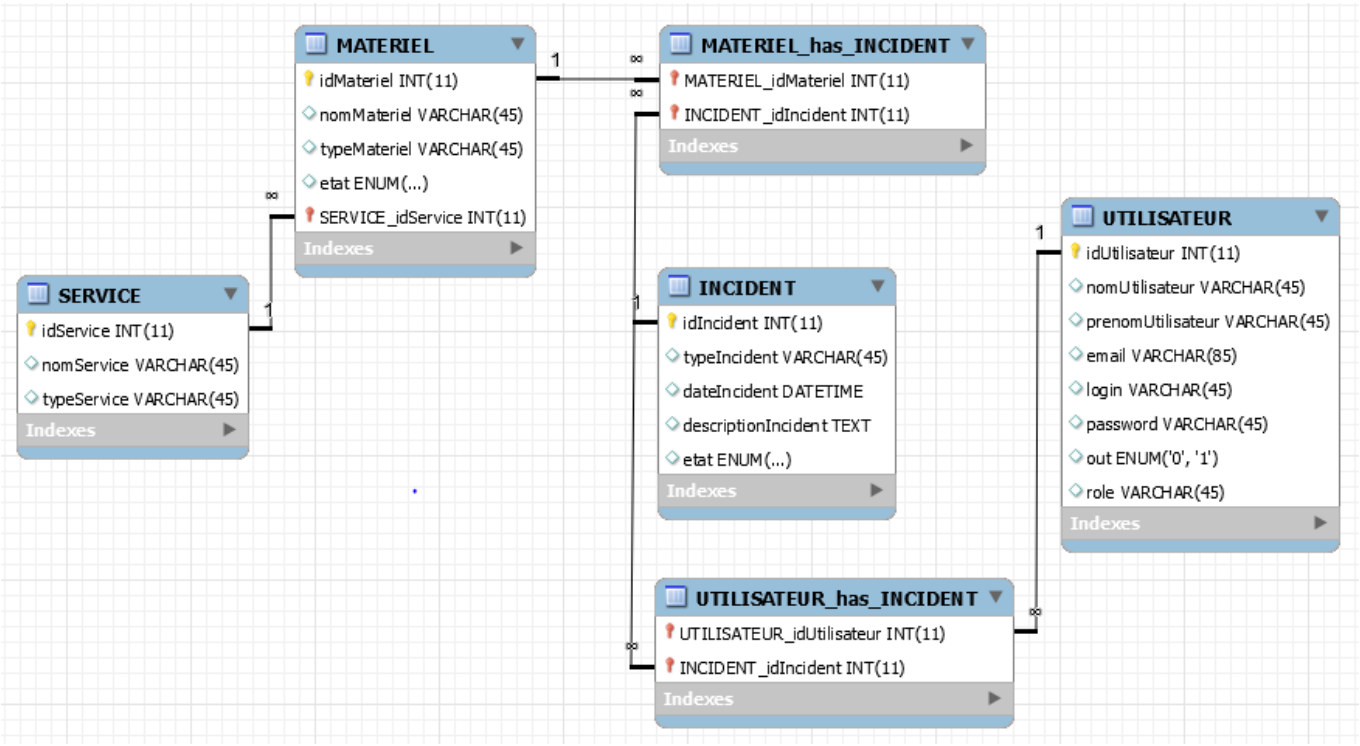
CE QUI EST ATTENDUE

- Une solution opérationnelle fonctionnant sous MYSQL
- Extraits d'implémentation contenant des illustrations : extrait code source, ...
- Notice d'utilisation destinée aux utilisateurs de l'application web.

B

ADELAIDE DONOVAN
CRETE JONATHAN

SCHEMA RELATIONNEL



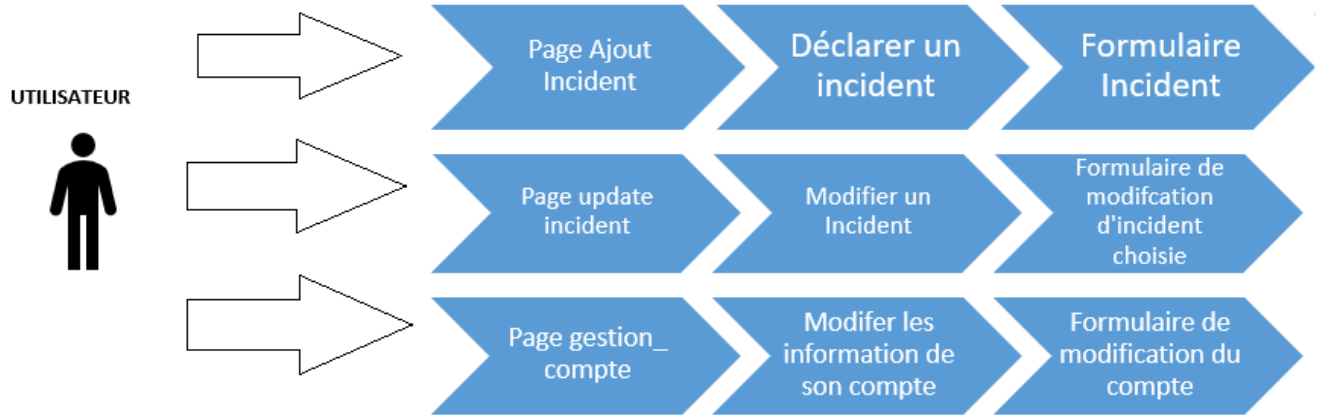
Nous Avons conçus la base de données « Dunord » a l'aide des objectifs et attentes du client Mr.Edgar Dunord, crée dans un premier temps sur le logiciel WorkBench « Logiciels de conception de base de données » et importer sous PhpMyadmin « application Web de gestion pour les systèmes de gestion de base de données MySQL »

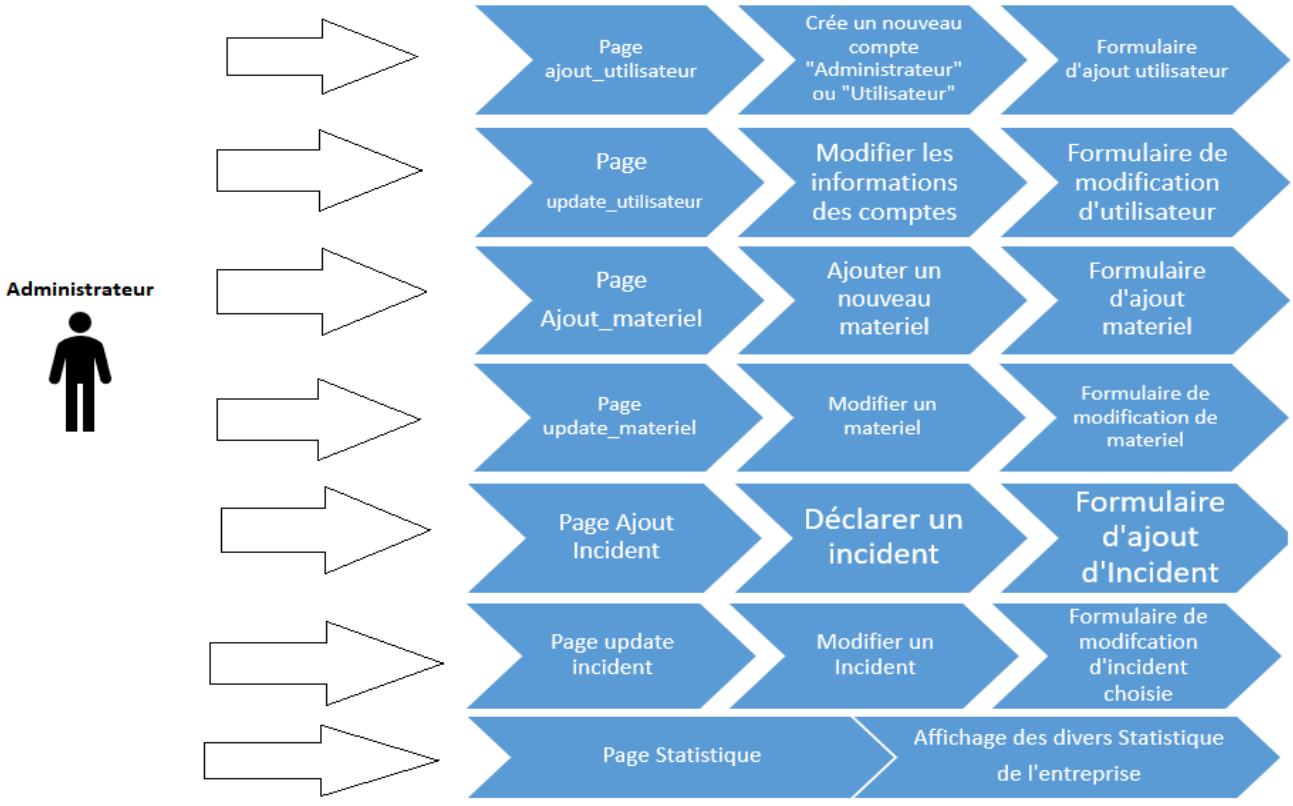
B

ADELAIDE DONOVAN
CRETE JONATHAN

DIAGRAMME DE CAS D'UTILISATION

Le diagramme de cas d'utilisation est un diagramme UML utilisés pour donner une vision globale du comportement fonctionnel d'un système.





B

ADELAIDE DONOVAN
CRETE JONATHAN

EXTRAIT D'IMPLEMENTATION

La majorité des fonctions ont été conçus dans un fichier nommé ''fonctions.php'' afin de pouvoir dissocier le FRONT-END ''HTML/CSS'' coté client du BACK-END ''PHP/PDO'' coté serveur. Chaque fonction est nommée et commenter afin de faciliter la reprise du projet pour une amélioration future.

Ci-dessus vous trouverez un extrait des fonction conçus dans le cadre de notre mission.

```php
/*
-------------------------------------------------------------------------
Fonction : getAuthentification
-------------------------------------------------------------------------
Description :
Permet de ce connecter avec son identifiant et mot de passe.
-------------------------------------------------------------------------
Arguments :
$login : Identifiant.
$pass : Mot de passe.
-------------------------------------------------------------------------
Retour : True si utilisateur et mot de passe correct, sinon False
------------------------------------------------------------------------- */
function getAuthentification($login, $pass) {
    global $pdo;
    $query = "SELECT * FROM UTILISATEUR WHERE login=:login and password=:pass";
    try {
        $prep = $pdo->prepare($query);
        $prep->bindValue(':login', $login);
        $prep->bindValue(':pass', $pass);
        $prep->execute();

        if($prep->rowCount() == 1){
            $result = $prep->fetch(PDO::FETCH_ASSOC);
            return $result;
        }
        else{
            return false;
        }
    }
    catch ( Exception $e ) {
        die ("Erreur dans la requete ".$e->getMessage());
    }
}
```

La fonction « getAuthentification » prend 2 argument en paramètre le login et le mot de passe de l'utilisateur si les informations rentrées par l'utilisateur sont présent dans la base de donnée elle récupère ses informations associées (nom, prénom, rôle, etc.. et le lui renvoie sous forme de tableau associatif

B

ADELAIDE DONOVAN
CRETE JONATHAN

Le rôle de cette fonction est de sécuriser les entrées sur l'application web (système de session).

```php
require_once('fonctions.php');
// on teste si nos variables sont définies et remplies
if (isset($_GET['login']) && isset($_GET['pwd']) ) {
// on appele la fonction getAuthentification en lui passant en paramètre le login et password
$result = getAuthentification($_GET['login'], $_GET['pwd']);

    // si le résulat est VRAI
    if($result != false){
        // on la démarre la session
        session_start ();
        // on enregistre les paramètres de notre visiteur comme variables de session
        $_SESSION['nom'] = $result['nomUtilisateur'];
        $_SESSION['prenom'] = $result['prenomUtilisateur'];
        $_SESSION['idUtilisateur'] = $result['idUtilisateur'];
        $_SESSION['email']=$result['email'];
        $_SESSION['role'] = $result['role'];
        $_SESSION['login'] = $result['login'];
        $_SESSION['password'] = $result['password'];
        $_SESSION['out'] = $result['out'];
        // on redirige notre visiteur vers une page de notre section membre
        header ('location: index.php');

    }
    //si le résultat est false on redirige vers la page d'authentification
    else{
        header ('location: authentification.php?erreur');
    }
}
//si nos variables ne sont pas renseignées on redirige vers la page d'authentification
else {
    header ('location: authentification.php?erreur');
}
```

Dans ce fichier de traitement de sessions que nous avons nommé « login.php » nous avons exploiter notre fonction « getAuthentification » elle permet de générer une session avec les informations de l'utilisateur récupérer par notre fonction « getAuthentification » si les condition de notre fonction ont bien été respecter « login et mot de passe présent dans la base de données » sinon l'utilisateur est automatiquement rediriger vers notre page d'authentification.

B

ADELAIDE DONOVAN
CRETE JONATHAN

Vous trouverez ci-dessous un extrait de notre page « authentification.php » (vue par le client)
Nous avons créé un formulaire classique (HTML) qui va permettre à l'utilisateur d'insérer des données que nous allons récupérer par la requête HTPP « GET » nécessaire à notre système de sessions.

```
<div class="container">
    <form action="login.php" method="GET" class="connexion">

    <!--  Champ d'insertion de l'identifiant  de l'utilisateur -->
        <label class="autentification">Votre login : </label><input id="autentification" type="text" name="login" placeholder="Identifiant"></p>

    <!--  Champ d'insertion du mot de passe   de l'utilisateur -->
        <label class="autentification">Votre mot de passe : </label><input id="autentification" type="password" name="pwd" placeholder="Mot de Passe"></p>


        <input class="log" type="submit" value="Se connecter">

</form>
```

C'est-à-dire que HTTP est une forme de langage qui va permettre au client (l'utilisateur, par le biais de son navigateur) de communiquer avec un serveur connecté au réseau (le serveur HTTP installé sur le serveur d'un site, par exemple Apache).

Dans le cadre de notre mission nous avons utilisé quatre opérations de base permettant le stockage, la mise à jour, la suppression d'informations dans notre base de données communément appelée CRUD (CREATE/READ/UPDATE/DELETE)

- CREATE : créer
- READ : lire
- UPDATE : mettre à jour
- DELETE : supprimer

```php
require_once('fonctions.php');

    //----- Recuperation des élément passer en url ------/

    $idUtilisateur = $_POST['idUtilisateur'];
    $nomUtilisateur = $_POST['nomUtilisateur'];
    $prenomUtilisateur= $_POST['prenomUtilisateur'];
    $login = $_POST['login'];
    $password = $_POST['password'];
    $email = $_POST['email'];
    $role = $_POST['role'];
    //----- Mise a jour d'un Utilisateur-----/
    $sql = "UPDATE UTILISATEUR SET
    nomUtilisateur = :nomUtilisateur,
    prenomUtilisateur = :prenomUtilisateur,
    login = :login,
    password = :password,
    email = :email,
    role = :role
    WHERE idUtilisateur = :idUtilisateur ";

    try {
    $stmt = $pdo->prepare($sql);
    $stmt->bindValue(':nomUtilisateur', $nomUtilisateur, PDO::PARAM_STR);
    $stmt->bindValue(':prenomUtilisateur', $prenomUtilisateur, PDO::PARAM_STR);
    $stmt->bindValue(':login', $login, PDO::PARAM_STR);
    $stmt->bindValue(':password', $password, PDO::PARAM_STR);
    $stmt->bindValue(':email', $email, PDO::PARAM_STR);
    $stmt->bindValue(':role', $role, PDO::PARAM_STR);
    $stmt->bindValue(':idUtilisateur',$idUtilisateur, PDO::PARAM_STR);
    $stmt->execute();
    }
catch ( Exception $e ) {
die ("Erreur dans la requete ".$e->getMessage());
}
```

Comme indiquer plutôt ci-dessus nous avons conçus des opérations de base permettant le stockage d'information dans notre base de données. Nous avons conçu des fichiers de traitement de donnée permettant à l'utilisateur de modifier, d'ajouter, de lire les données présentes dans la base.

Dans cet extrait nous récupérons les données de l'utilisateur passée par la méthode post (requête http) afin de modifier ses informations selon son identifiant (unique présent dans les données enregistré par notre système de session lors de la création de la session de l'utilisateur).

B

ADELAIDE DONOVAN
CRETE JONATHAN

Nous avons établi une gestion de rôle utilisateur dans le cadre de notre mission afin de sécuriser l'accès à des données dites sensible (partit administrateur).

Comme vous le remarquerez dans le menue de notre application web, seul une session en cours ayant comme rôle « Admin » peuvent accéder à toutes les ressources du site.

Dans le cas où une personne malveillante essaierait de contourner notre sécurité en passant par l'url il sera directement redigérer vers notre page d'authentification.

```
        <!-- Nav -->
        <nav id="nav">
            <ul>
                <li class="current"><a href="index.php">Accueil</a></li>
                <li><a href="ajout_incident.php">Ajout d'un incident</a></li>
                <li><a href="liste_incident.php">Liste incidents</a></li>
            <?php  if(($_SESSION['role'] =="Admin")){ ?>
                <li><a href="liste_materiel.php">Liste Materiel</a></li>
                <li><a href="ajout_materiel.php">Ajout Materiel</a></li>
                <li><a href="statistique.php">Statistique(s)</a></li>
                <?php } ?>
            </ul>
        </nav>

    </div>
</div>
```

Lors de la conception des différents fonctions, page de traitements de notre application web nous avons gérer les erreurs afin de tester notre application avant sa livraison à l'aide du bloc Try catch, si une erreur se produit lors de l'exécution du code présent dans le bloc Try, PHP va exécuter le code qui se trouve dans le catch, permettant ainsi de gérer l'erreur.

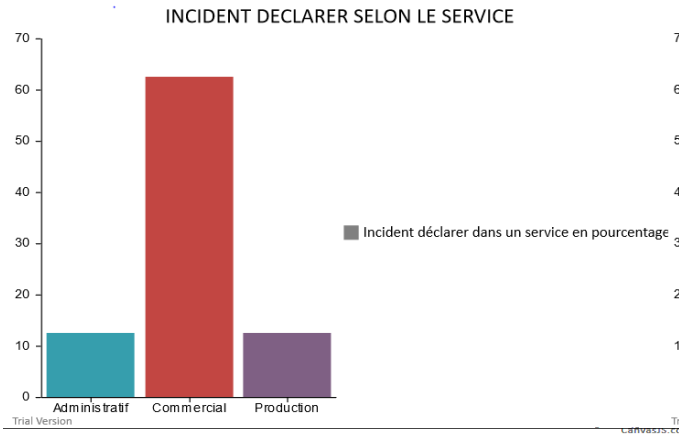
```
function getAllUser(){
    global $pdo;

      $query = 'SELECT * FROM UTILISATEUR;' ;
    try {
        $result = $pdo->query($query)->fetchAll(PDO::FETCH_ASSOC);
    return $result;

        }
    catch ( Exception $e ) {
        die ("Erreur dans la requete ".$e->getMessage());
    }
    }
```

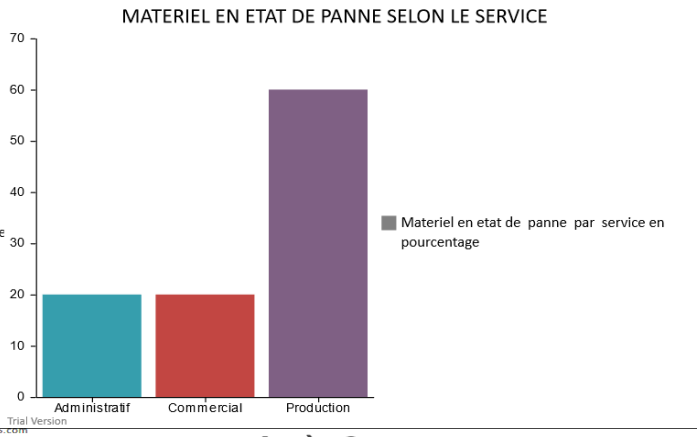
B

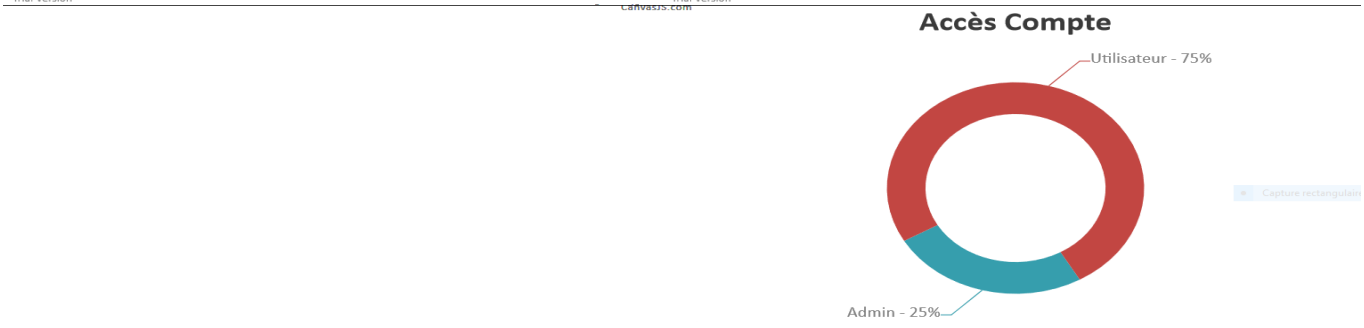
ADELAIDE DONOVAN
CRETE JONATHAN

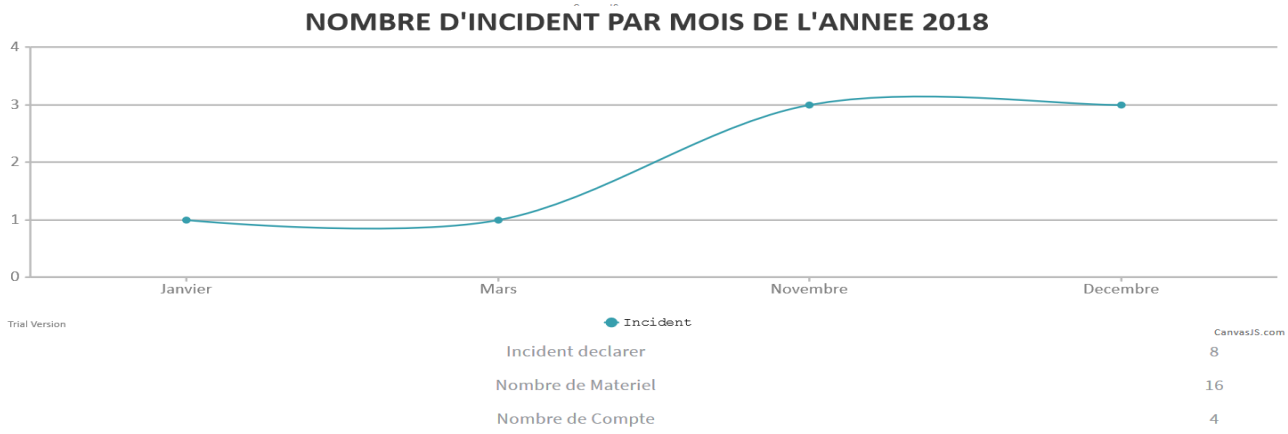
Afin d'apporter une réelle plus-value lors de la conception de notre application web nous avons exploiter un plugin javascript « CanvasJs » permettant la création de divers graphiques voyez ci-dessous.

A l'aide de CanvasJs nous avons conçus une application web évolutive permettant de gérer les incidents ainsi que les utilisateur sous forme de graphique dans le temps (nombre d'ordinateur en état de marche en temp réeel, nombre d'incident déclarer par mois sur l'année , ect..)









Incident declarer	8
Nombre de Materiel	16
Nombre de Compte	4

B

ADELAIDE DONOVAN
CRETE JONATHAN

FUTURES EVOLUTIONS DU PROJET DUNORD

- Passage de la solution actuelle sous un modèle MVC (modèle vue Controller)
- Intégration de la solution actuelle sous un Framework (Laravel,Symfony)
- Optimisation du code existant
- Ajout future de l'application web sur un service d'hébergement