Documentation Technique M2L

1. Description générale
2. La base de données
3. L’API REST

1. Description générale de l’application

Nom du site : Maison des Ligues

Présentation : L’application de la Maison des Ligues est une application de gestion électronique qui permet aux employés et administrateurs de superviser et gérer les produits. Avec une interface conviviale et intuitive, il offre une expérience de gestion optimale.

Fonctionnalités principales :

- **Dashboard Admin - Gestion des Produits** : le dashboard offre aux administrateurs un ensemble d'outils puissants pour gérer efficacement les produits disponibles sur la plateforme.

Les administrateurs peuvent facilement ajouter de nouveaux produits à la plateforme en remplissant un formulaire détaillé. Ils peuvent spécifier des informations telles que le nom du produit, la description, les images, les prix, les quantités disponibles, etc.

Les administrateurs ont la possibilité de modifier les détails des produits existants à tout moment. Ils peuvent mettre à jour des informations telles que le prix, la description, les images, la disponibilité, etc.

En cas de besoin, les administrateurs peuvent supprimer des produits de la plateforme. Cette fonctionnalité est utile pour retirer les produits discontinués ou obsolètes de la liste des produits disponibles.

2. La base de données

La base de données est composée de trois tables :

Table : compte

Description : cette table stocke les informations des utilisateurs enregistrés sur la plateforme en afin les authentifier pour leurs futures interactions

```sql
CREATE TABLE IF NOT EXISTS `compte` (
  `uuid` varchar(32) NOT NULL,
  `prenom` varchar(50) DEFAULT NULL,
  `nom` varchar(50) DEFAULT NULL,
  `email` varchar(50) DEFAULT NULL,
  `mdp` varchar(100) DEFAULT NULL,
  `estadmin` int(11) DEFAULT NULL,
  PRIMARY KEY (`uuid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

Table : produit

Description : Cette table stocke les informations sur les produits disponibles à l'achat afin d'afficher les produits sur le site, gérer les stocks et effectuer des transactions d'achat.

```sql
CREATE TABLE IF NOT EXISTS `produit` (
  `uuid` varchar(32) NOT NULL,
  `nom` varchar(50) DEFAULT NULL,
  `description` varchar(50) DEFAULT NULL,
  `prix` int(11) DEFAULT NULL,
  `quantite` int(11) DEFAULT NULL,
  `visibilite` int(11) DEFAULT NULL,
  `image` varchar(100) DEFAULT NULL,
  PRIMARY KEY (`uuid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

Table : panier

Description : cette table stocke les détails des paniers d'achat des utilisateurs et est liée aux deux tables précédentes.

```sql
CREATE TABLE IF NOT EXISTS `panier` (
  `id` int(11) NOT NULL,
  `uuid_compte` varchar(32) DEFAULT NULL,
  `uuid_produit` varchar(32) DEFAULT NULL,
  `quantite` int(11) DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `FK_panier_produit` (`uuid_produit`),
  KEY `FK_panier_compte` (`uuid_compte`),
  CONSTRAINT `FK_panier_compte` FOREIGN KEY (`uuid_compte`) REFERENCES `compte` (`uuid`) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `FK_panier_produit` FOREIGN KEY (`uuid_produit`) REFERENCES `produit` (`uuid`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

3. L'API Rest

L'application communique avec la base de données via une API créée avec NodeJS et Express.

On utilise un middleware servant à authentifier les personnes connectés grâce à un jeton de connexion temporaire avec JsonWebToken.

```
function verifierJWT(req, res, next) {
    const token = req.headers.authorization.substring(7);
    if (!token) {
        return res.status(401).json({ message: 'Pas de token, accès non autorisé' });
    }

    try {
        const decoded = jwt.verify(token, secretKey);
        req.user = decoded;
        next();
    } catch (error) {
        return res.status(403).json({ message: 'Token invalide' });
    }
}
```

La gestion des comptes

```
> app.post('/inscription', async(req,res)=>{ ···
  });

> app.post('/inscription/mail', async(req,res)=>{ ···
  });

> app.post('/connexion', async(req,res)=>{ ···
  });

> app.get('/autorisation/:token', verifierJWT, async(req,res)=>{ ···
  });
```

La gestion des produits

```
> app.get('/produit', async(req,res)=>{ ···
  });

> app.get('/produit/:uuid', verifierJWT, async(req,res)=>{ ···
  });

> app.delete('/produit/:uuid', verifierJWT, async(req,res)=>{ ···
  });

> app.post('/produit/panier', verifierJWT, async(req,res)=>{  ···
  });
```

La gestion du panier

```
app.get('/panier/:token', verifierJWT, async(req,res)=>{ ···
});

app.delete('/panier/compte/:token', verifierJWT, async(req,res)=>{ ···
});

app.delete('/panier/compte/produit/:idPanier', async(req,res)=>{ ···
});

app.put('/panier/validation/:token', verifierJWT, async(req,res)=>{ ···
});

app.get('/historique/:token', verifierJWT, async(req,res)=>{ ···
});
```

La gestion du dashboard administrateur

```
app.get('/dashboard/produits',verifierJWT, async(req,res)=>{ ···
});

app.get('/dashboard/utilisateurs',verifierJWT, async(req,res)=>{ ···
});

app.delete('/john', async (req, res) => { ···
});

app.post('/dashboard/ajouterproduit', verifierJWT, upload.single("image"), async (req, res) => { ···
});

app.put('/dashboard/modifierproduit/:uuid', verifierJWT, async(req,res)=>{ ···
});
```