L'objectif est de rendre votre application Next.js accessible en ligne, et de la connecter à une base de données PostgreSQL hébergée et gérée par Neon, ce qui est une excellente combinaison pour un projet Next.js et Prisma.

Procédure de Déploiement Détaillée (Étape par Étape)

Nous allons procéder en trois parties :

1. Mettre en place votre base de données PostgreSQL sur Neon.
2. Préparer votre projet Next.js localement pour le déploiement.
3. Déployer votre application Next.js sur Vercel.

Partie 1 : Création et configuration de la base de données PostgreSQL sur Neon

Neon offre une base de données PostgreSQL sans serveur, idéale pour les applications Next.js.

1. **Créer un compte Neon :**
    - Ouvrez votre navigateur et allez sur https://neon.tech/.
    - Cliquez sur "Sign Up" ou "Get Started Free".
    - Vous pouvez vous inscrire avec votre compte GitHub, Google ou par e-mail. Choisissez l'option la plus rapide pour vous.
2. **Créer un nouveau projet Neon :**
    - Une fois connecté, Neon vous guidera pour créer un nouveau projet.
    - Donnez un **Nom de projet** (par exemple, BusinessBook-DB).
    - Choisissez une **Région** qui est géographiquement proche de vos utilisateurs (ou de vous-même pour les tests).
    - Choisissez la **version de PostgreSQL** (laissez par défaut ou sélectionnez la version la plus récente compatible avec votre schema.prisma si vous l'avez spécifié, mais les versions sont généralement rétrocompatibles).
    - Cliquez sur "Create Project".
3. **Récupérer la chaîne de connexion (Connection String) :**
    - Après la création du projet, Neon affichera les détails de votre base de données.
    - **La chose la plus importante ici est la "Connection String" (chaîne de connexion).** Elle ressemblera à quelque chose comme : postgresql://user:password@ep-random-name-12345.us-east-1.aws.neon.tech/dbname?sslmode=require
    - **Copiez cette chaîne de connexion complète.** C'est elle que votre application utilisera pour se connecter à la base de données. Gardez-la en lieu sûr, elle contient des informations sensibles (utilisateur, mot de passe).

4. **Optionnel mais recommandé : Migrer vos données existantes (si nécessaire) :**
    - Si votre BusinessBook s'appuie sur des données déjà présentes dans votre base de données locale (utilisateurs, produits, etc.), vous devrez migrer ces données vers votre nouvelle base de données Neon.
    - **Méthode simplifiée pour la présentation :**
        - Dans pgAdmin, faites un clic droit sur votre base de données locale, sélectionnez "Backup...". Sauvegardez-la au format "Plain" et cochez "Data only" si vous avez déjà un schéma sur Neon (via Prisma), sinon "Schema and data".
        - Dans DBeaver, connectez-vous à votre base de données Neon (utilisez la chaîne de connexion obtenue à l'étape 3).
        - Ensuite, dans DBeaver, faites un clic droit sur la base de données Neon, et cherchez une option "Restore..." ou "Execute Script..." pour exécuter le fichier .sql que vous avez exporté depuis pgAdmin.
    - **Pour un déploiement plus robuste :** Utilisez les outils de migration de données de PostgreSQL (par exemple, pg_dump et pg_restore) ou les outils intégrés de Neon si disponibles. Pour une présentation rapide, une restauration via DBeaver est souvent suffisante si le volume est faible.

Partie 2 : Préparation de votre projet Next.js localement pour le déploiement

Vous devez indiquer à votre projet comment se connecter à la nouvelle base de données Neon.

1. **Mettre à jour le fichier .env.local :**
    - À la racine de votre projet BusinessBook (là où se trouve package.json), créez ou ouvrez le fichier .env.local.
    - Ajoutez ou mettez à jour la variable d'environnement DATABASE_URL avec la chaîne de connexion Neon que vous avez copiée précédemment.
    - Assurez-vous également que la variable AUTH_SECRET est bien définie dans ce fichier, comme vous l'avez fait précédemment.

<!-- end list -->

Extrait de code

DATABASE_URL="postgresql://user:password@ep-random-name-12345.us-east-1.aws.neon.tech/dbname?sslmode=require"

AUTH_SECRET="VOTRE_CLE_SECRETE_GENEREE_POUR_AUTHJS"

Ajoutez d'autres variables d'environnement si votre projet en utilise

    - **Important :** Les guillemets autour de la chaîne de connexion sont obligatoires si elle contient des caractères spéciaux.

2. **Mettre à jour prisma/schema.prisma (si nécessaire) :**
    - Ouvrez le fichier prisma/schema.prisma.
    - Vérifiez que la configuration de votre datasource pointe vers la variable d'environnement DATABASE_URL. Elle devrait ressembler à ceci :

<!-- end list -->

Extrait de code

```
datasource db {
  provider = "postgresql"
  url      = env("DATABASE_URL")
}
```

    - Si vous aviez une URL locale en dur, assurez-vous qu'elle est remplacée par env("*DATABASE_URL*").
3. **Générer et pousser le schéma Prisma vers Neon :**
    - Ouvrez votre terminal dans VS Code (à la racine de votre projet).
    - Exécutez la commande suivante pour générer le client Prisma avec la nouvelle configuration de base de données :

Bash

```
>> pnpm prisma generate
```

    - Ensuite, poussez votre schéma de base de données (vos modèles Prisma) vers la base de données Neon. Cela créera les tables nécessaires sur Neon si elles n'existent pas encore.

Bash

```
>>pnpm prisma db push
```

        - **Note :** db push est rapide et convient pour les phases de développement et les démos. Pour la production avec des données existantes, pnpm prisma migrate deploy est généralement utilisé après avoir généré des migrations, mais pour 2 heures, db push est plus direct si le schéma est simple.
4. **Vérifier le .gitignore :**
    - Assurez-vous que votre fichier .gitignore (également à la racine du projet) contient .env.local pour éviter de le pousser sur GitHub. C'est essentiel pour la sécurité.

5. **Pousser les dernières modifications sur GitHub :**
    - Comme votre camarade a déjà poussé les dernières modifications, assurez-vous que les modifications liées à .env.local (le fichier lui-même NE DOIT PAS être poussé, mais son contenu est utilisé localement) et prisma/schema.prisma (si modifié) sont bien dans votre dépôt GitHub.
    - Si vous avez fait des changements locaux (comme prisma/schema.prisma ou ajouté un .env.local pour les tests locaux), faites un commit et push vers votre dépôt GitHub :

Bash

```
>> git add .
>> git commit -m "Configure for Vercel deployment with Neon"
>> git push origin main # ou le nom de votre branche principale
```

Partie 3 : Déploiement de l'application Next.js sur Vercel

Maintenant que votre base de données est prête et votre projet configuré, nous allons déployer sur Vercel.

1. **Connectez-vous à Vercel :**
    - Ouvrez votre navigateur et allez surhttps://vercel.com/.
    - Cliquez sur "Log in" ou "Sign Up".
    - **Connectez-vous avec votre compte GitHub.** C'est la méthode la plus simple et la plus recommandée, car Vercel se liera directement à vos dépôts GitHub.
2. **Importer votre projet Git :**
    - Une fois connecté, vous verrez probablement un tableau de bord.
    - Cliquez sur "Add New..." ou le bouton "New Project".
    - Vercel vous demandera d'importer un dépôt Git.
    - Sélectionnez votre compte GitHub. Vercel listera vos dépôts.
    - Trouvez et **sélectionnez le dépôt businessbook** que votre camarade a poussé.
    - Cliquez sur "Import".
3. **Configurer le projet sur Vercel :**
    - Vercel détectera automatiquement que c'est une application **Next.js**. Le "Framework Preset" devrait être réglé sur "Next.js".
    - **"Root Directory" :** Normalement, cela devrait être laissé vide si votre package.json est à la racine du dépôt. Si votre projet Next.js est dans un sous-dossier, indiquez ce sous-dossier ici.

- **"Build and Output Settings" :** Laissez les valeurs par défaut (Vercel connaît les commandes de build pour Next.js : npm run build ou pnpm run build si votre projet est configuré pour pnpm).
    - Vérifiez que le "Build Command" est bien pnpm build et l'"Install Command" est pnpm install (Vercel est intelligent et le détectera avec votre pnpm-lock.yaml).
- **Configuration des Variables d'Environnement (C'EST LE PLUS IMPORTANT) :**
    - Sous la section "Environment Variables", vous devez ajouter les variables que vous avez mises dans votre .env.local (mais pas le fichier lui-même !).
    - Cliquez sur "Add Row" et ajoutez :
        - **Name:** DATABASE_URL
        - **Value:** La chaîne de connexion complète de Neon (celle que vous avez copiée à l'étape 3 de la Partie 1).
        - **Name:** AUTH_SECRET
        - **Value:** Votre clé secrète générée pour next-auth.
    - **Assurez-vous de ne pas faire d'erreur de frappe.**

4. **Déployer :**
    - Cliquez sur le bouton **"Deploy"**.
    - Vercel va maintenant récupérer votre code depuis GitHub, installer les dépendances, construire votre application et la déployer.
    - Vous verrez un écran de "Building" et "Deploying". Patientez.
5. **Vérifier le déploiement :**
    - Une fois le déploiement terminé, Vercel vous fournira une URL (par exemple, https://businessbook-xxxx.vercel.app).
    - Cliquez sur cette URL pour ouvrir votre application Next.js déployée.