Projet de SGBD
Ahmed Ltayef Info2 Groupe B

1-creation des tables

```
CREATE TABLE articles(

    refart CHAR(4) PRIMARY KEY,
    designation VARCHAR2(30) NOT NULL,
    prixA NUMBER(8,2) NOT NULL,
    prixV NUMBER(8,2) NOT NULL,
    codetva NUMBER(1) NOT NULL,
    categorie CHAR(10),
    qtestk NUMBER(5) NOT NULL,
    supp CHAR(1) DEFAULT 'N' CHECK (supp IN ('Y','N'))

);
```

Sortie de script | Tâche terminée en 0,033 secondes

```
Table ARTICLES créé(e).
```

```
ALTER TABLE articles
ADD CONSTRAINT verif_prixV_prixA CHECK (prixV > prixA);
```

Sortie de script | Tâche terminée en 0,03 secondes

```
Table ARTICLES modifié(e).
```

```
CREATE TABLE clients (
    noclt       NUMBER(4) CONSTRAINT clients_pk PRIMARY KEY,
    nomclt      VARCHAR2(30) NOT NULL,
    prenomclt   VARCHAR2(30) NULL,
    adrclt      VARCHAR2(60) NOT NULL,
    code_postal NUMBER(5) NOT NULL,
    villeclt    VARCHAR2(30) NOT NULL,
    telclt      VARCHAR2(8) NOT NULL,
    adrmail     VARCHAR2(60) NULL,

    CONSTRAINT clients_code_postal_ck CHECK (code_postal BETWEEN 1000 AND 99999)


);
```

Sortie de script | Tâche terminée en 0,029 secondes

```
Table CLIENTS créé(e).
```

```
CREATE TABLE Postes(codeposte VARCHAR(10) CONSTRAINT poste_pk PRIMARY KEY,
libelle    VARCHAR2(30) NOT NULL,
indice NUMBER(2) NOT NULL);
```

Sortie de script

Tâche terminée en 0,028 secondes

```
Table POSTES créé(e).
```

```
CREATE TABLE personnel (
    idpers       NUMBER(4) CONSTRAINT personnel_pk PRIMARY KEY,
    nompers      VARCHAR2(30) NOT NULL,
    prenompers   VARCHAR2(30) NOT NULL,
    adrpers      VARCHAR2(60) NOT NULL,
    villepers    VARCHAR2(30) NOT NULL,
    telpers      VARCHAR2(8) NOT NULL,
    d_embauche   DATE NOT NULL,
    login        VARCHAR2(30)  UNIQUE NOT NULL,
    motP         CHAR(8) NOT NULL,
    codeposte    VARCHAR2(10) NOT NULL,



    CONSTRAINT fk_personnel_poste FOREIGN KEY (codeposte) REFERENCES postes (codeposte)
);
```

Sortie de script

Tâche terminée en 0,027 secondes

```
Table PERSONNEL créé(e).
```

```
CREATE TABLE commandes (
    nocde     NUMBER(6) CONSTRAINT pk_commandes PRIMARY KEY,
    noclt     NUMBER(4) NOT NULL,
    datecde   DATE DEFAULT SYSDATE ,
    etatcde   CHAR(2) DEFAULT 'EC' NOT NULL,

    CONSTRAINT fk_commandes_clients FOREIGN KEY(noclt) REFERENCES clients(noclt),
    CONSTRAINT check_commandes_etat CHECK (etatcde IN ('EC','PR','LI','SO','AN','AL'))
);
```

Sortie de script

Tâche terminée en 0,029 secondes

```
Table COMMANDES créé(e).
```

```
CREATE TABLE ligcdes(
    nocde NUMBER(6),
    refart CHAR(4) NOT NULL,
    qtecde NUMBER(5) NOT NULL,
    CONSTRAINT fk_ligcdes_commandes FOREIGN KEY(nocde) REFERENCES commandes(nocde),
    CONSTRAINT fk_ligcdes_articles FOREIGN KEY(refart) REFERENCES articles(refart),
    CONSTRAINT pk_ligcdes PRIMARY KEY(nocde,refart));
```

Sortie de script

Tâche terminée en 0,024 secondes

```
Table LIGCDES créé(e).
```

```
CREATE TABLE LivraisonCom(
    nocde NUMBER(6) NOT NULL,
    dateliv DATE NOT NULL,
    livreur  NUMBER(4) NOT NULL,
    modepay CHAR(10) NOT NULL,
    etatliv CHAR(2) NOT NULL,
    CONSTRAINT fk_Livraison_com FOREIGN KEY (nocde)REFERENCES commandes(nocde),
    CONSTRAINT fk_livraison_pers FOREIGN KEY(livreur) REFERENCES personnel(idpers),
    CONSTRAINT ck_livraison_modep CHECK(modepay IN('avant_livraison' , 'apres_livraison')),
    CONSTRAINT pk_livraison_com PRIMARY KEY(nocde, dateliv),
    CONSTRAINT ck_livraison_com_etat CHECK (etatliv IN ('EC','LI','AL'))
    );
```

Sortie de script

Tâche terminée en 0,027 secondes

```
Table LIVRAISONCOM créé(e).
```

```
CREATE TABLE HCommandesAnnulees(
    nocde NUMBER(6) NOT NULL,
    numclt NUMBER(4) NOT NULL,
    nbrart NUMBER(5) NOT NULL,
    montantc NUMBER(10,2) NOT NULL,
    datecde DATE NOT NULL,
    dateAnnulation DATE DEFAULT SYSDATE,
    code_postal NUMBER(5) NOT NULL,
    AvantLiv CHAR(1) DEFAULT 'N' NOT NULL,
    CONSTRAINT pk_hcmd_annul PRIMARY KEY (nocde,numclt),
    CONSTRAINT fk_hcmd_nocde FOREIGN KEY (nocde) REFERENCES commandes(nocde),
    CONSTRAINT fk_hca_clients FOREIGN KEY (numclt) REFERENCES clients(noclt),
    CONSTRAINT ck_hca_avantliv CHECK (AvantLiv IN ('Y','N'))
);
```

Sortie de script

Tâche terminée en 0,03 secondes

```
Table HCOMMANDESANNULEES créé(e).
```

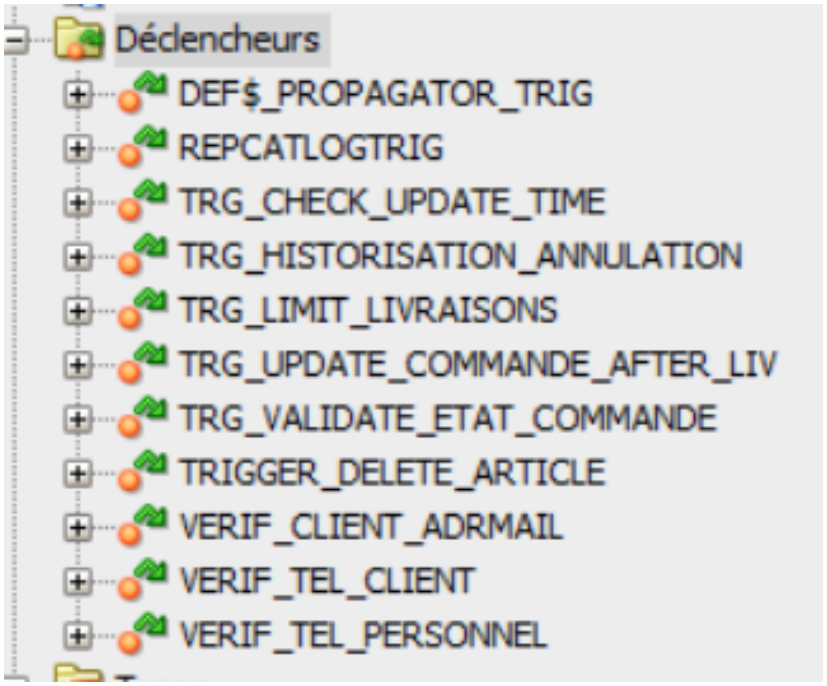
2-sequences/Index

```
CREATE SEQUENCE seq_clients
START WITH 1
INCREMENT BY 1
NOCACHE
NOCYCLE;

CREATE SEQUENCE seq_commandes
START WITH 1
INCREMENT BY 1
NOCACHE
NOCYCLE;

CREATE SEQUENCE seq_personnel
START WITH 1
INCREMENT BY 1
NOCACHE
NOCYCLE;

CREATE SEQUENCE seq_articles_ref
START WITH 1
INCREMENT BY 1
NOCACHE
NOCYCLE;


CREATE SEQUENCE seq_poste
START WITH 1
INCREMENT BY 1
NOCACHE
NOCYCLE
```

```
Sequence SEQ_CLIENTS créé(e).

Sequence SEQ_COMMANDES créé(e).

Sequence SEQ_PERSONNEL créé(e).

Sequence SEQ_ARTICLES_REF créé(e).

Sequence SEQ_POSTE créé(e).
```

```
CREATE INDEX idx_clients_nom ON clients(nomclt);

CREATE INDEX idx_commandes_date ON commandes(datecde);
CREATE INDEX idx_commandes_numero ON commandes(nocde);
CREATE INDEX idx_commandes_client ON commandes(noclt);

CREATE INDEX idx_article_categorie ON articles(categorie);
CREATE INDEX idx_articles_code ON articles(refart);
CREATE INDEX idx_articles_designation ON articles(designation);

CREATE INDEX idx_livraison_livreur ON LivraisonCom(livreur);
CREATE INDEX idx_livraison_date ON LivraisonCom(dateliv);
CREATE INDEX idx_livraison_commande ON LivraisonCom(nocde);

CREATE INDEX idx_personnel_login ON personnel(login);

CREATE INDEX idx_ligcdes_article ON ligcdes(refart);
```

3-Triggers




```
create or replace TRIGGER verif_tel_personnel
BEFORE INSERT OR UPDATE ON personnel
FOR EACH ROW
BEGIN
    IF NOT REGEXP_LIKE(:NEW.telpers, '^[0-9]{8}$') THEN
        RAISE_APPLICATION_ERROR(
            -20004,
            'Le numéro de téléphone doit comporter exactement 8 chiffres (0-9).'
        );
    END IF;
END;
```

```
create or replace TRIGGER verif_tel_client
BEFORE INSERT OR UPDATE ON clients
FOR EACH ROW
BEGIN
    IF NOT REGEXP_LIKE(:NEW.telclt, '^[0-9]{8}$') THEN
        RAISE_APPLICATION_ERROR(
            -20004,
            'Le numéro de téléphone doit comporter exactement 8 chiffres (0-9).'
        );
    END IF;
END;
```

```
create or replace TRIGGER verif_client_adrmail
BEFORE INSERT OR UPDATE ON clients
FOR EACH ROW
DECLARE
    v_count NUMBER;
BEGIN
    SELECT COUNT(*)
    INTO v_count
    FROM clients
    WHERE adrmail = :NEW.adrmail;


    IF v_count > 0 THEN
        RAISE_APPLICATION_ERROR(-20002, 'client deja existe');
    END IF;
END;
```

```
create or replace TRIGGER trigger_delete_article
BEFORE DELETE ON articles
FOR EACH ROW
DECLARE
    v_count NUMBER;
BEGIN
    SELECT COUNT(*) INTO v_count
    FROM ligcdes
    WHERE refart = :OLD.refart;

    IF v_count > 0 THEN
        UPDATE articles
        SET supp = 'Y'
        WHERE refart = :OLD.refart;

        RAISE_APPLICATION_ERROR(
            -20012,
            '. Suppression logique '
        );
    END IF;
END;
```

```
create or replace TRIGGER trg_validate_etat_commande
BEFORE UPDATE OF etatcde ON commandes
FOR EACH ROW
BEGIN
    -- EC = En cours
    IF :OLD.etatcde = 'EC' THEN
        IF :NEW.etatcde NOT IN ('PR', 'AN') THEN
            RAISE_APPLICATION_ERROR(-20010,
                'Transition non autorisée : EC -> ' || :NEW.etatcde);
        END IF;
    END IF;

    -- PR = Prête
    IF :OLD.etatcde = 'PR' THEN
        -- LI, AN sont autorisés. AL n'est pas autorisé directement ici,
        -- il doit passer par l'annulation de la livraison.
        IF :NEW.etatcde NOT IN ('LI', 'AN', 'AL') THEN
            RAISE_APPLICATION_ERROR(-20010,
                'Transition non autorisée : PR -> ' || :NEW.etatcde);
        END IF;
    END IF;

    -- LI = Livrée
    IF :OLD.etatcde = 'LI' THEN
        IF :NEW.etatcde != 'SO' THEN
            RAISE_APPLICATION_ERROR(-20010,
                'Transition non autorisée : LI -> ' || :NEW.etatcde);
        END IF;
    END IF;

    -- États terminaux : aucun changement permis
    IF :OLD.etatcde IN ('SO', 'AN', 'AL') THEN
        RAISE_APPLICATION_ERROR(-20010,
            'Impossible de modifier une commande ' || :OLD.etatcde);
    END IF;

END;
```

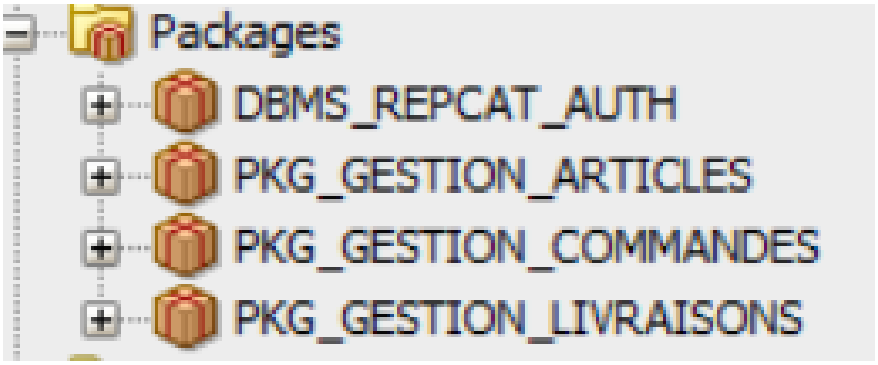
```
create or replace TRIGGER trg_update_commande_after_liv

AFTER UPDATE OF etatliv ON LivraisonCom
FOR EACH ROW
BEGIN
    IF :NEW.etatliv = 'AL' AND :OLD.etatliv != 'AL' THEN
        UPDATE commandes
        SET etatcde = 'AL'
        WHERE nocde = :NEW.nocde;
    END IF;
END;
```

```plsql
create or replace TRIGGER trg_limit_livraisons
BEFORE INSERT OR UPDATE OF livreur, dateliv ON LivraisonCom
FOR EACH ROW
DECLARE
    v_count NUMBER;
    v_code_postal clients.code_postal%TYPE;
BEGIN
    -- ✕ Ne rien faire lors d'une UPDATE
    IF UPDATING THEN
        RETURN;
    END IF;

    -- ✓ Seulement pour INSERT
    SELECT cl.code_postal INTO v_code_postal
    FROM commandes c
    JOIN clients cl ON c.noclt = cl.noclt
    WHERE c.nocde = :NEW.nocde;

    SELECT COUNT(*) INTO v_count
    FROM LivraisonCom lc
    JOIN commandes c ON lc.nocde = c.nocde
    JOIN clients cl ON c.noclt = cl.noclt
    WHERE lc.livreur = :NEW.livreur
      AND TRUNC(lc.dateliv) = TRUNC(:NEW.dateliv)
      AND cl.code_postal = v_code_postal
      AND lc.etatliv != 'AL';

    IF v_count >= 15 THEN
        RAISE_APPLICATION_ERROR(
            -20006,
            'Limite de 15 livraisons par jour et par code postal atteinte pour ce livreur.'
        );
    END IF;
END;
```

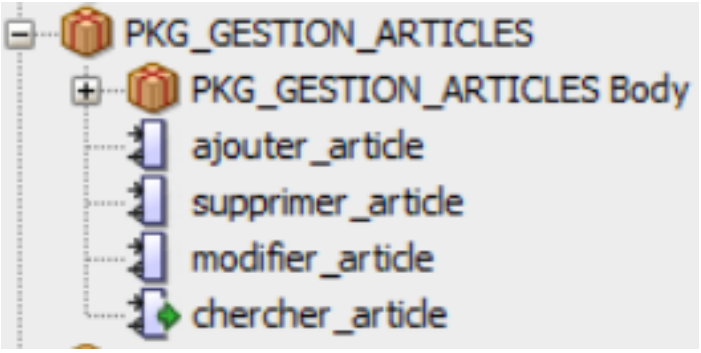
```plsql
create or replace TRIGGER trg_historisation_annulation
AFTER UPDATE OF etatcde ON commandes
FOR EACH ROW
WHEN (NEW.etatcde IN ('AN', 'AL') AND OLD.etatcde NOT IN ('AN', 'AL'))
DECLARE
    v_nbr_art NUMBER;
    v_montant NUMBER;
    v_avant_liv CHAR(1);
BEGIN
    -- Calculer le nombre d'articles
    SELECT COUNT(*) INTO v_nbr_art
    FROM ligcdes
    WHERE nocde = :OLD.nocde;

    -- Calculer le montant
    SELECT SUM(l.qtecde * a.prixV) INTO v_montant
    FROM ligcdes l
    JOIN articles a ON l.refart = a.refart
    WHERE l.nocde = :OLD.nocde;

    -- Déterminer si avant livraison
    -- Si une ligne existe dans LivraisonCom, c'est 'N' (après livraison)
    SELECT CASE WHEN COUNT(*) > 0 THEN 'N' ELSE 'Y' END INTO v_avant_liv
    FROM LivraisonCom
    WHERE nocde = :OLD.nocde;

    -- Insérer dans l'historique
    INSERT INTO HCommandesAnnulees (
        nocde, numclt, nbrart, montantc, datecde,
        code_postal, avantLiv
    )
    SELECT :OLD.nocde, c.noclt, v_nbr_art, NVL(v_montant, 0),
           :OLD.datecde, cl.code_postal, v_avant_liv
    FROM commandes c
    JOIN clients cl ON c.noclt = cl.noclt
    WHERE c.nocde = :OLD.nocde;

EXCEPTION
```

```
create or replace TRIGGER trg_check_update_time
BEFORE UPDATE ON LivraisonCom
FOR EACH ROW
DECLARE
    v_current_hour NUMBER;
    v_livraison_hour NUMBER;
BEGIN
    IF :OLD.dateliv != :NEW.dateliv OR :OLD.livreur != :NEW.livreur THEN
        v_current_hour := TO_NUMBER(TO_CHAR(SYSDATE, 'HH24'));
        v_livraison_hour := TO_NUMBER(TO_CHAR(:NEW.dateliv, 'HH24'));
        IF v_livraison_hour < 14 THEN
            IF v_current_hour >= 9 THEN
                RAISE_APPLICATION_ERROR(-20008,
                    'Modification impossible après 9h pour les livraisons du matin.');
            END IF;
        ELSE
            IF v_current_hour >= 14 THEN
                RAISE_APPLICATION_ERROR(-20009,
                    'Modification impossible après 14h pour les livraisons de l''après-midi.');
            END IF;
        END IF;
    END IF;
END;
```

4-Packages

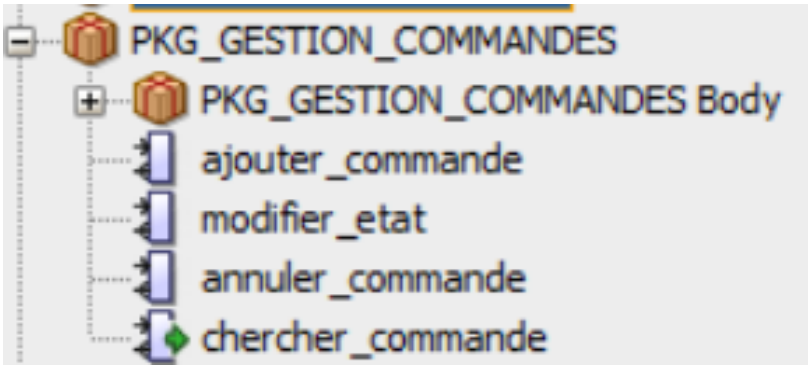


4.1-Package articles




```
create or replace PACKAGE pkg_gestion_articles AS

    -- Procédures principales
    PROCEDURE ajouter_article(
        p_designation IN VARCHAR2,
        p_prix_achat IN NUMBER,
        p_prix_vente IN NUMBER,
        p_codetva IN NUMBER,
        p_categorie IN VARCHAR2 DEFAULT NULL,
        p_qtestk IN NUMBER DEFAULT 0,
        p_refart OUT CHAR  -- CORRIGÉ : Changé de VARCHAR2 à CHAR
    );

    PROCEDURE supprimer_article(p_refart IN CHAR);  -- CORRIGÉ : Changé de VARCHAR2 à CHAR

    PROCEDURE modifier_article(
        p_refart IN CHAR,  -- CORRIGÉ : Changé de VARCHAR2 à CHAR
        p_designation IN VARCHAR2 DEFAULT NULL,
        p_prix_achat IN NUMBER DEFAULT NULL,
        p_prix_vente IN NUMBER DEFAULT NULL,
        p_codetva IN NUMBER DEFAULT NULL,
        p_categorie IN VARCHAR2 DEFAULT NULL,
        p_qtestk IN NUMBER DEFAULT NULL
    );

    FUNCTION chercher_article(
        p_refart IN CHAR DEFAULT NULL,  -- CORRIGÉ : Changé de VARCHAR2 à CHAR
        p_designation IN VARCHAR2 DEFAULT NULL,
        p_categorie IN VARCHAR2 DEFAULT NULL
    ) RETURN SYS_REFCURSOR;

END pkg_gestion_articles;
```

4.2-Package commandes




```
create or replace PACKAGE BODY pkg_gestion_commandes AS

    PROCEDURE ajouter_commande(p_noclt IN NUMBER, p_nocde OUT NUMBER) IS
    BEGIN
        SELECT seq_commandes.NEXTVAL INTO p_nocde FROM dual;
        INSERT INTO commandes(nocde, noclt, datecde, etatcde)
        VALUES (p_nocde, p_noclt, SYSDATE, 'EC');
    END ajouter_commande;

   PROCEDURE modifier_etat(p_nocde IN NUMBER, p_nouvel_etat IN CHAR) IS
    v_etat_actuel commandes.etatcde%TYPE;
    v_count_liv NUMBER;
    v_article_count NUMBER;
BEGIN
    SELECT etatcde INTO v_etat_actuel FROM commandes WHERE nocde = p_nocde;
     -- Si transition vers PR, vérifier qu'il y a au moins 1 article
    IF p_nouvel_etat = 'PR' THEN
        SELECT COUNT(*) INTO v_article_count FROM ligcdes WHERE nocde = p_nocde;
        IF v_article_count = 0 THEN
            RAISE_APPLICATION_ERROR(-20015, 'La commande doit contenir au moins un article pour passer en état Prête (PR).');
        END IF;
    END IF;

    -- Vérifier si une livraison existe pour cette commande
    SELECT COUNT(*) INTO v_count_liv FROM LivraisonCom WHERE nocde = p_nocde;

    -- Logique de transition :
    -- EC -> PR, AN
    -- PR -> LI, AN, AL
    -- LI -> SO
    IF (v_etat_actuel = 'EC' AND p_nouvel_etat IN ('PR','AN')) OR
       (v_etat_actuel = 'PR' AND p_nouvel_etat IN ('LI','AN','AL')) OR
       (v_etat_actuel = 'LI' AND p_nouvel_etat = 'SO') THEN

        -- Règle 1: Si état actuel = PR et une livraison existe, on ne peut pas passer à AN
        IF v_etat_actuel = 'PR' AND p_nouvel_etat = 'AN' AND v_count_liv > 0 THEN
            RAISE_APPLICATION_ERROR(-20013, 'Impossible d''annuler une commande PR qui a une livraison associée.');
        END IF;
```

```
        SELECT etatcde INTO v_etat FROM commandes WHERE nocde = p_nocde;

        -- Vérifier si une livraison existe pour cette commande
        SELECT COUNT(*) INTO v_livraison_existe
        FROM LivraisonCom
        WHERE nocde = p_nocde;

        -- 1. Vérification des états terminaux
        IF v_etat IN ('LI','SO','AN','AL') THEN
            RAISE_APPLICATION_ERROR(-20011, 'Impossible d''annuler une commande dans cet état (' || v_etat || ').');
        END IF;

        -- 2. Logique d'annulation :
        IF v_livraison_existe > 0 THEN
            -- Si une livraison existe, l'annulation doit se faire via la livraison (AL)
            RAISE_APPLICATION_ERROR(-20013, 'Annulation impossible. Une livraison existe. Veuillez annuler la livraison via pkg_gestion_livraisons.
        ELSE
            -- Si aucune livraison n'existe, on passe à AN (Annulée Avant livraison)
            UPDATE commandes SET etatcde = 'AN' WHERE nocde = p_nocde;
            -- Le trigger trg_historisation_annulation gère l'insertion dans HCommandesAnnulees
        END IF;

    END annuler_commande;

    FUNCTION chercher_commande(p_nocde IN NUMBER DEFAULT NULL,
                               p_noclt IN NUMBER DEFAULT NULL,
                               p_date  IN DATE DEFAULT NULL) RETURN SYS_REFCURSOR IS
        rc SYS_REFCURSOR;
    BEGIN
        OPEN rc FOR
            SELECT * FROM commandes
            WHERE (p_nocde IS NULL OR nocde = p_nocde)
              AND (p_noclt IS NULL OR noclt = p_noclt)
              AND (p_date IS NULL OR TRUNC(datecde) = TRUNC(p_date));
        RETURN rc;
    END chercher_commande;

END pkg_gestion_commandes;
```

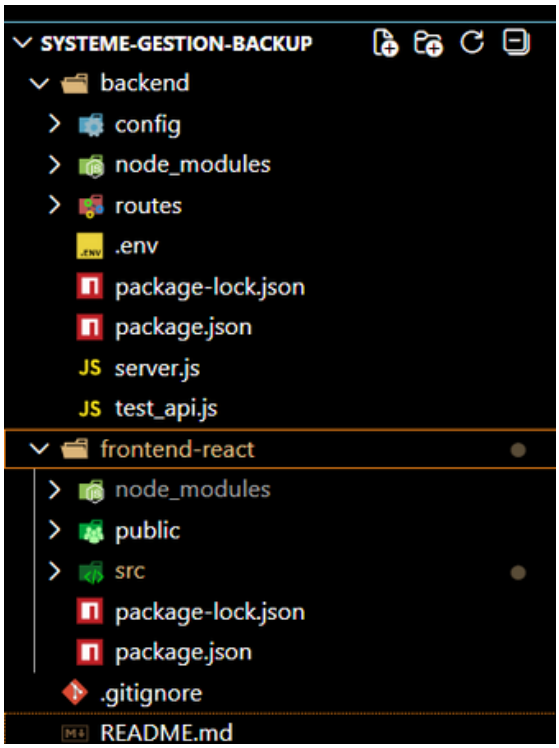
4.3-Package livraisons


```sql
create or replace PACKAGE pkg_gestion_livraisons AS
    -- Constante pour la limite de livraisons par jour et par code postal
    g_nb_max_livraisons CONSTANT NUMBER := 15;

    -- Vérifie si une commande est prête à être livrée (etatcde = 'PR')
    FUNCTION verifier_commande_pret(p_nocde IN NUMBER) RETURN BOOLEAN;
    -- Compte le nombre de livraisons d'un livreur pour une journée et un code postal
    FUNCTION compter_livraisons_livreur(
        p_livreur IN NUMBER,
        p_date IN DATE,
        p_code_postal IN NUMBER -- Utilisation du code postal
    ) RETURN NUMBER;
    -- Ajoute une nouvelle livraison
    PROCEDURE ajouter_livraison(
        p_nocde IN NUMBER,
        p_livreur IN NUMBER,
        p_dateliv IN DATE,
        p_modepay IN VARCHAR2 DEFAULT 'avant_livraison'
    );
    -- Annule une livraison (suppression logique : etatliv = 'AL')
    PROCEDURE supprimer_livraison(p_nocde IN NUMBER, p_dateliv IN DATE);
    -- Modifie la date et/ou le livreur d'une livraison
    PROCEDURE modifier_livraison(
        p_nocde IN NUMBER,
        p_dateliv IN DATE,
        p_nouvelle_date IN DATE DEFAULT NULL,
        p_nouveau_livreur IN NUMBER DEFAULT NULL
    );
    FUNCTION chercher_livraison(
        p_nocde IN NUMBER DEFAULT NULL,
        p_livreur IN NUMBER DEFAULT NULL,
        p_code_postal IN NUMBER DEFAULT NULL,
        p_date IN DATE DEFAULT NULL
    ) RETURN SYS_REFCURSOR;

END pkg_gestion_livraisons;
```

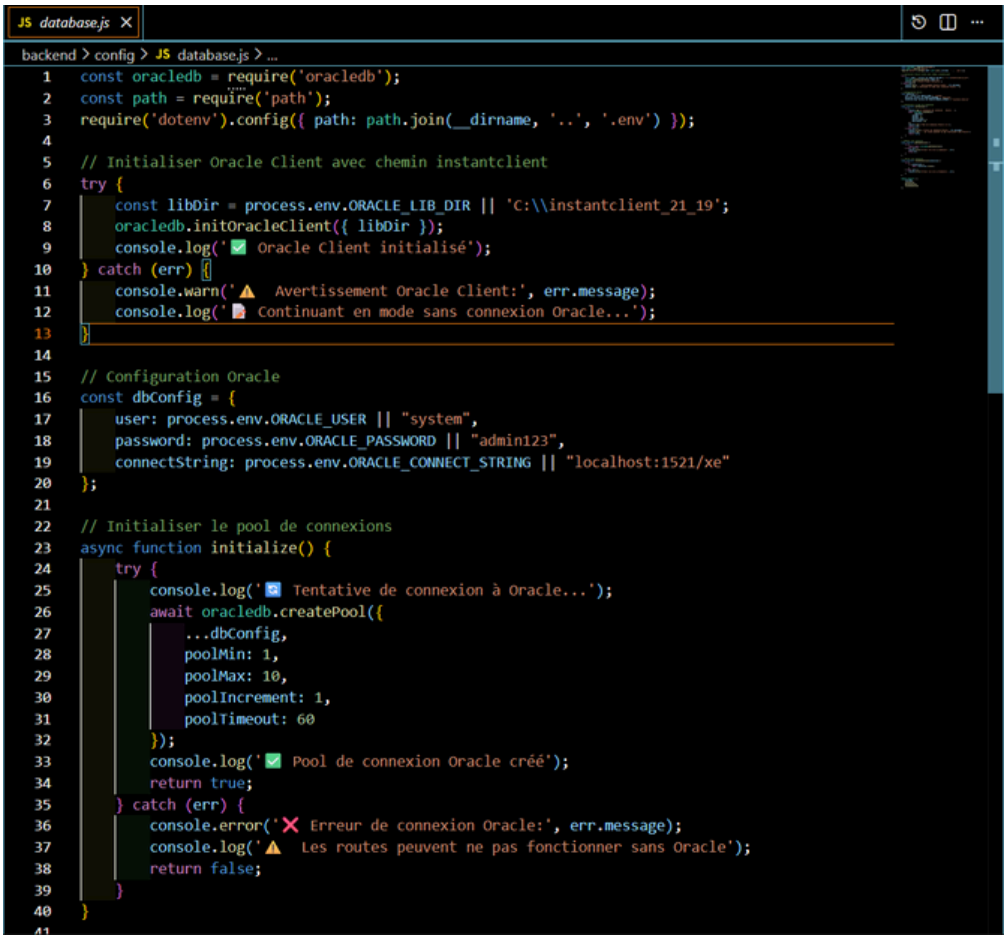
5-Interfaces

J'ai utilisé Node.js et Express pour le développement côté serveur, et React pour le développement de l'interface utilisateur

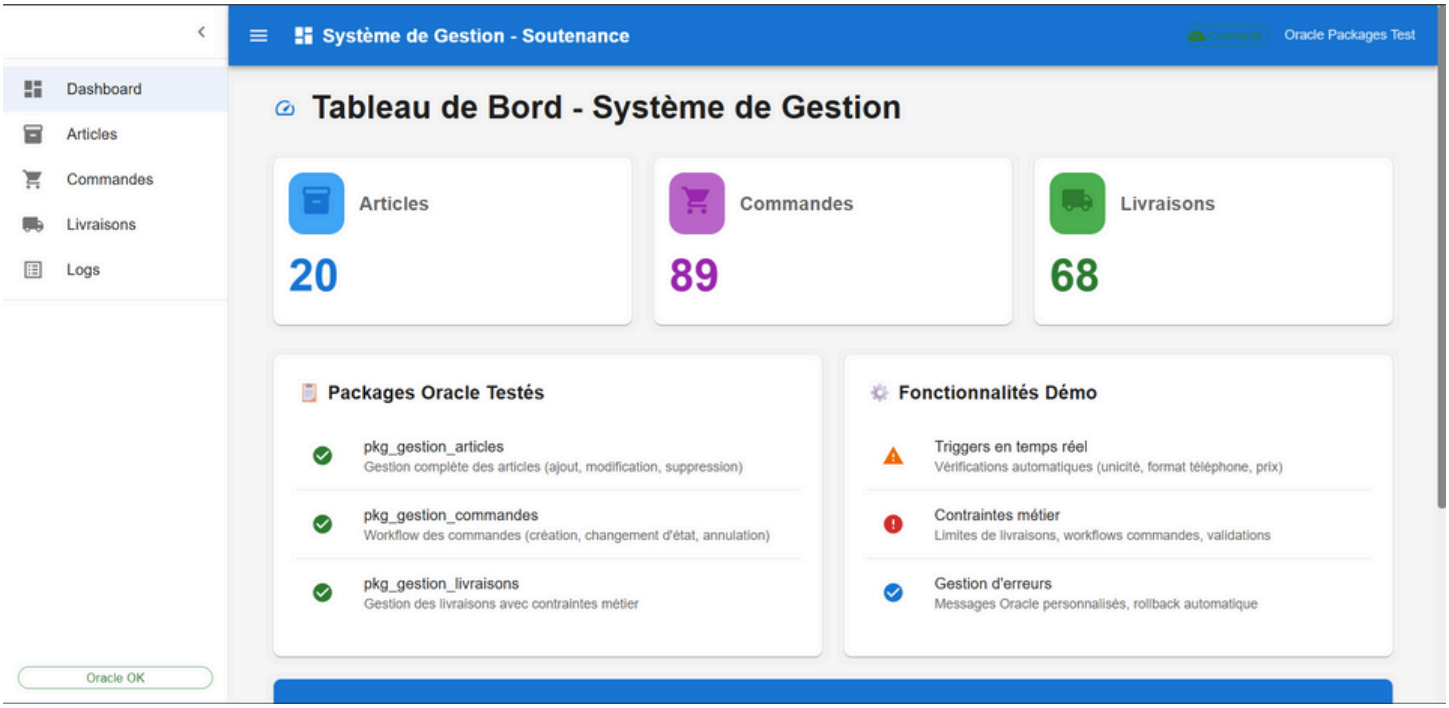
-Structure du projet-



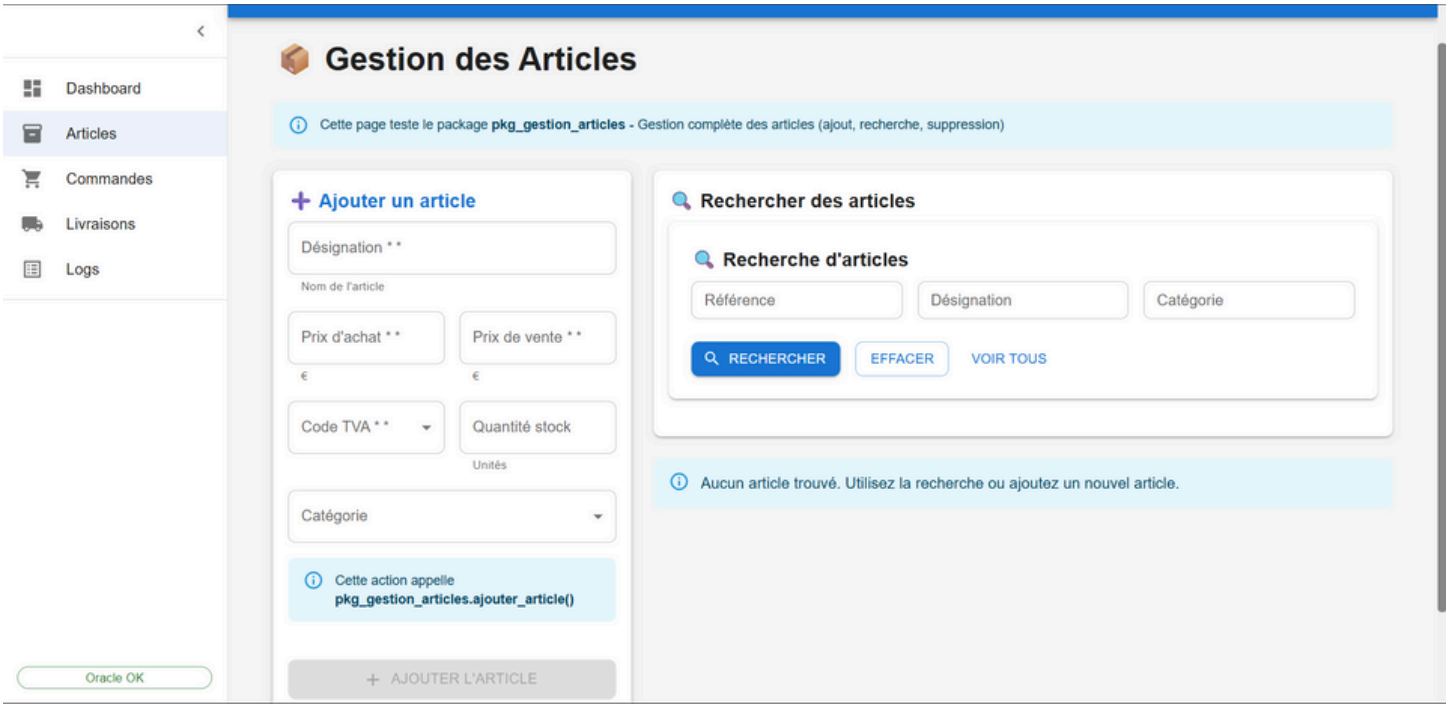
-page de connexion avec oracle-



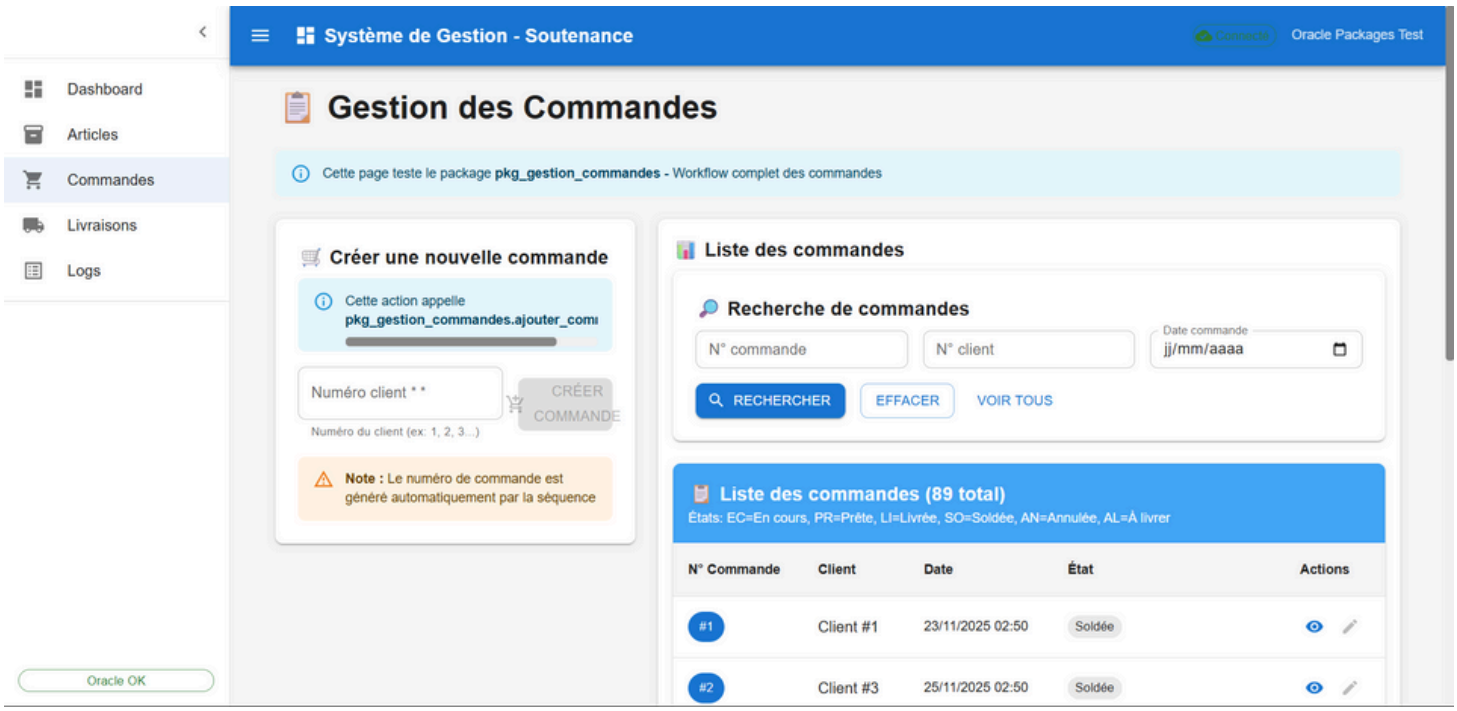
-Dashboard-



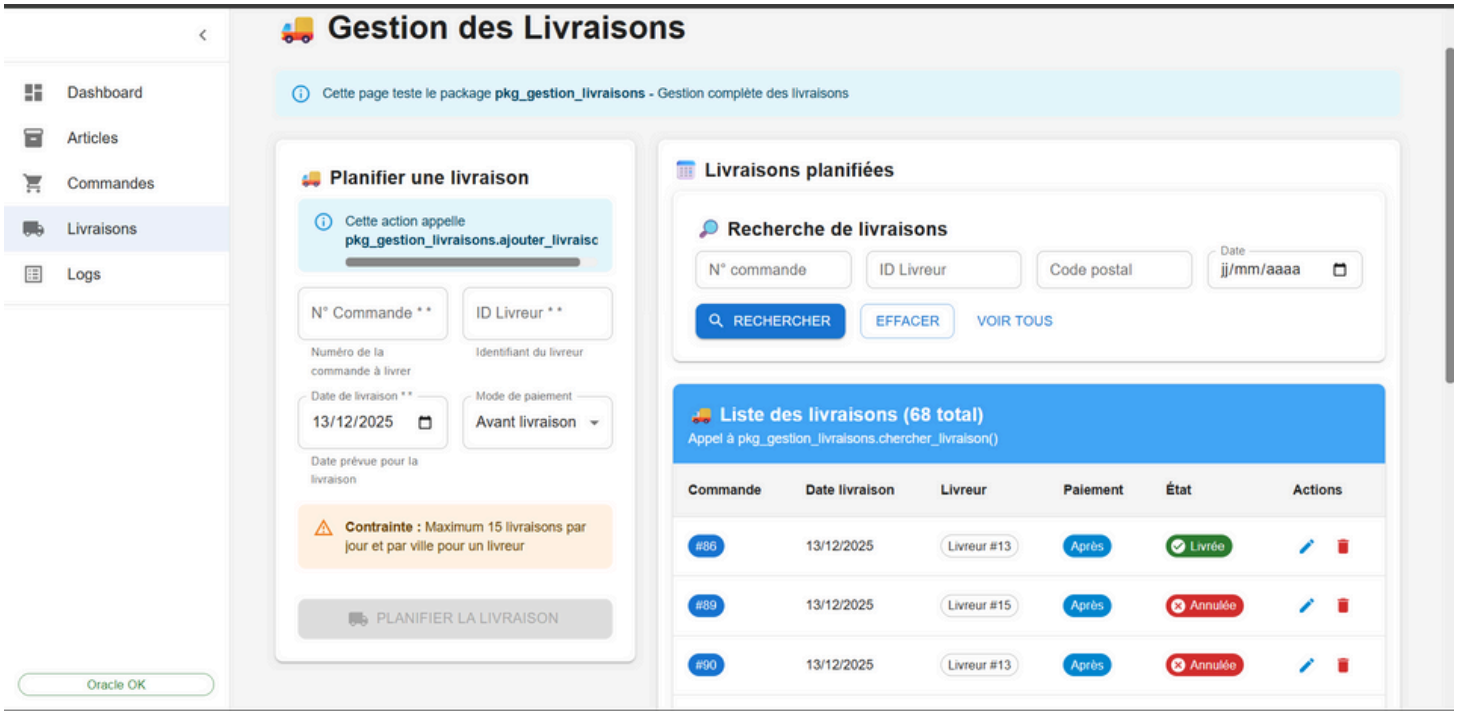
-Gestion des articles-



-Gestion des commandes-

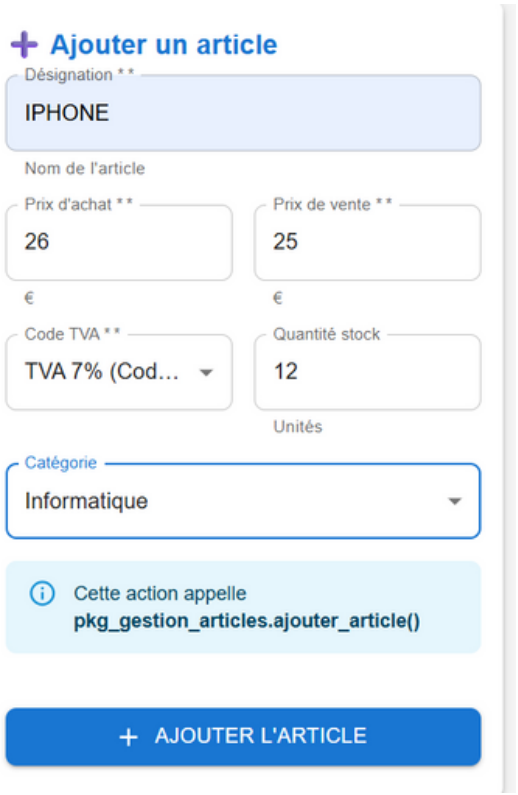


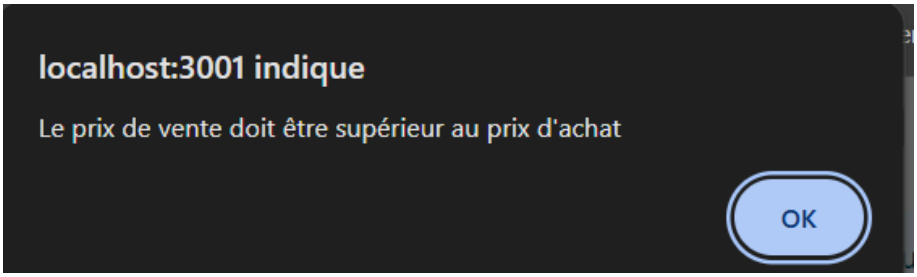
-Gestion des livraisons-



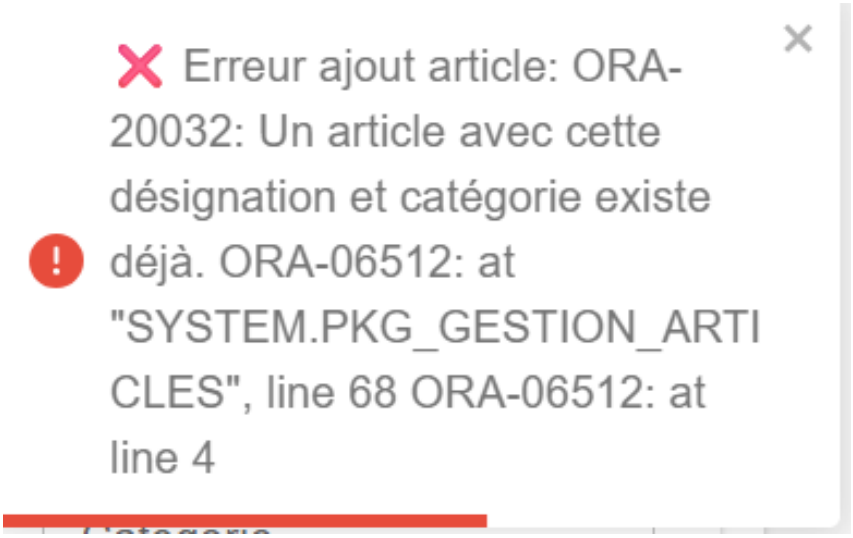
-Quelques Tests-

-Ajout d'un article avec prix de vente inférieur à prix d'achat

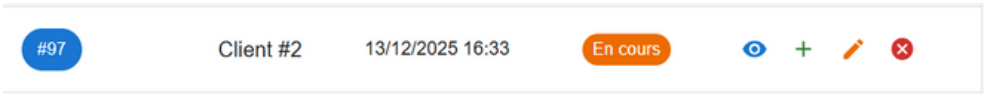




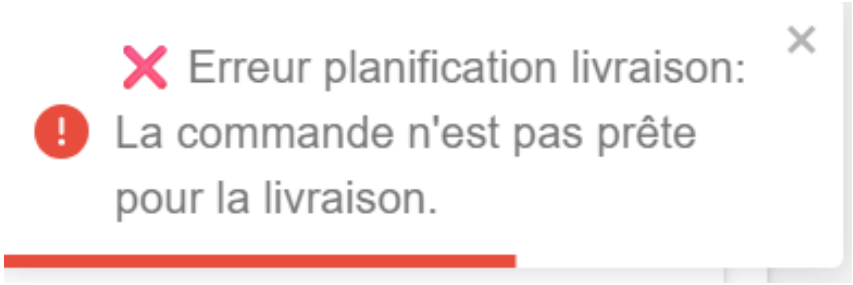
-Ajout d'un article qui existe déjà-



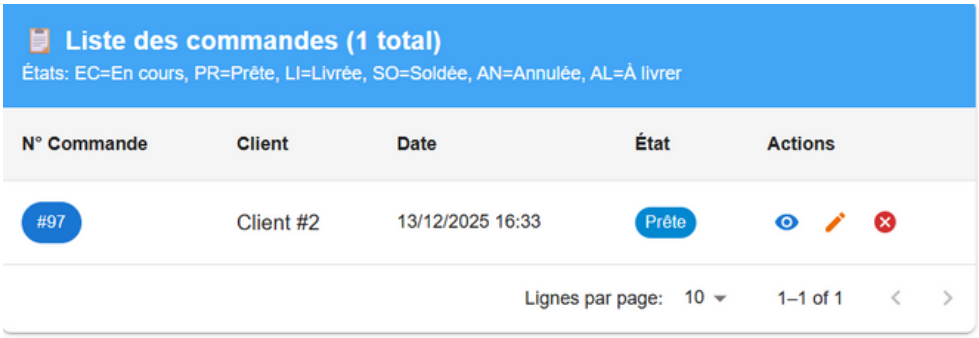
-commande crée avec état initial en cours-

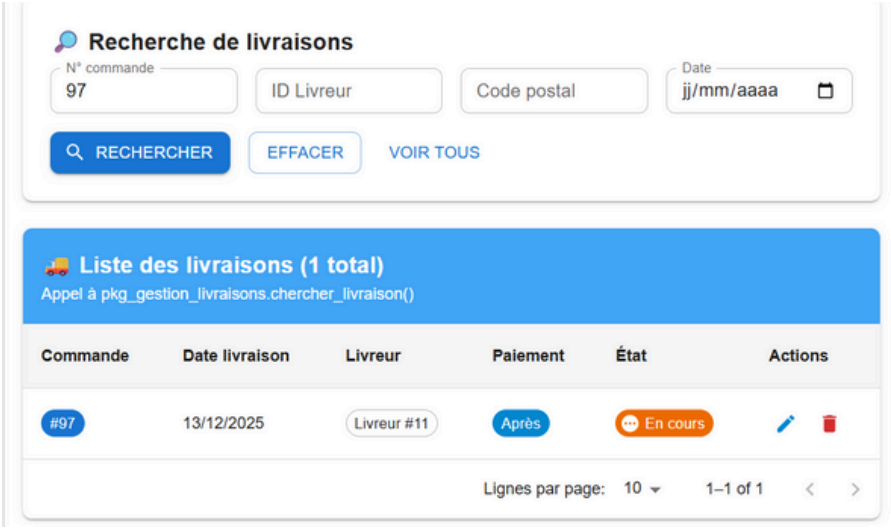


-commande doit etre prete pour passer au livraison-

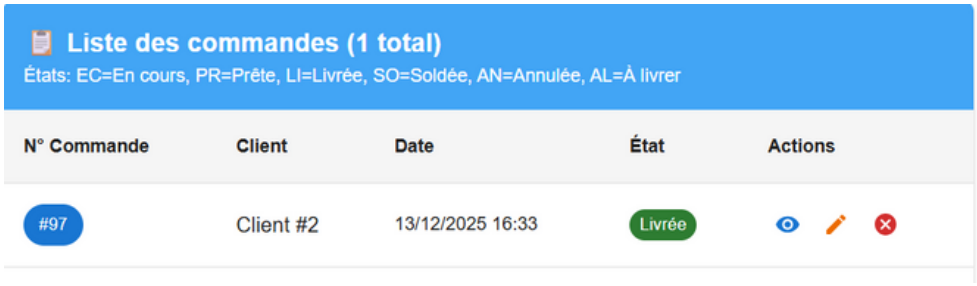


-commande prete et ajoutéé au livraison-





-synchronisation si commande passe à livrée-



-commande deja livrée ne peut etre supprimée-

-commande peut passer à l'etat soldée après etre livrée-