**Manuel d'utilisation de LARSON**

( Le progrAmme des adheRents à la Salle de MusculatiON )

**Fonctionnement du programme LARSON**

LARSON est un programme Java , en mode console, permettant le CRUD :

Create : Créer, Lire : Read , Mettre a jour : Updater, Supprimer : Delete ,

sur des informations dans une base de données , de stocker des objets, ainsi que leurs propriétés à l'aide du logiciel PgAdmin4.

Lors du lancement du programme LARSON via Eclipse , 6 options s'offrent a vous :



*1) Ajoutez un nouvel adherent :* cette option vous permet d'ajouter un nouvel adherent à la base de donnée. Un numero d'identification unique lui sera attribué automatiquement , nous devrez ensuite renseigner son nom , son age , sa taille , son poids . L'engregistrement se fera automatiquement ensuite sur la base de données.

*2) Consultez la liste des adherents* : cette option vous permet de consulter l'ensemble des adherents présents dans la base de données de la salle du musculation.

*3) Mise a jour d'un adherent :* cette option vous permets de mettre à jour les données deja saisie sur un adherent grace a son numero d'identification unique , par exemple le poids , ou l'age . La mise a jour se fera immédiatement et automatiquement sur la base de données.

*4) Supprimez un adherent* : cette option vous permet de supprimer un adherent de la base de donnée de la salle de musculation, en renseignant son numéro d'identifiant unique . La suppréssion sera effective immédiatement sur la base de données apres votre validation .

*5) Affichez les Perfs Max des adherents :* cette option vous permet d'afficher les performances maximum (entrées directement dans la base de données afin d'eviter toute modification de performance malveillante de la part des adherents. Ne pas tricher.)

*Pour toute autre touche :* cette option vous fait directement quitter le programme LARSON .

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Manuel de deploiment**

Pour la Table "MuscuEval" :

CREATE Scrpit :

-- Table: public."MuscuEval"

-- DROP TABLE public."MuscuEval";

CREATE TABLE public."MuscuEval"

(

"idCustom" integer NOT NULL DEFAULT nextval('"MuscuEval\_idCustom\_seq"'::regclass),

"nameCustom" character varying(50) COLLATE pg\_catalog."default" NOT NULL,

"ageCustom" integer NOT NULL,

"heightCustom" double precision NOT NULL,

"weightCustom" integer NOT NULL,

CONSTRAINT "MuscuEval\_pkey" PRIMARY KEY ("idCustom")

)

WITH (

OIDS = FALSE

)

TABLESPACE pg\_default;

ALTER TABLE public."MuscuEval"

OWNER to postgres;

DELETE Script :

DELETE FROM public."MuscuEval"

WHERE <condition>;

INSERT Script :

INSERT INTO public."MuscuEval"(

"idCustom", "nameCustom", "ageCustom", "heightCustom", "weightCustom")

VALUES (?, ?, ?, ?, ?);

SELECT Script :

SELECT "idCustom", "nameCustom", "ageCustom", "heightCustom", "weightCustom"

FROM public."MuscuEval";

UPDATE Script :

UPDATE public."MuscuEval"

SET "idCustom"=?, "nameCustom"=?, "ageCustom"=?, "heightCustom"=?, "weightCustom"=?

WHERE <condition>;

\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Pour la Table "Record" :

CREATE Scrpit :

-- Table: public."Record"

-- DROP TABLE public."Record";

CREATE TABLE public."Record"

(

"idPerf" integer NOT NULL DEFAULT nextval('"Record\_idPerf\_seq"'::regclass),

"idCustom" integer NOT NULL,

"maxBench" integer NOT NULL,

"maxSquat" integer NOT NULL,

"datePerf" integer NOT NULL,

CONSTRAINT "Record\_pkey" PRIMARY KEY ("idPerf"),

CONSTRAINT "idCustom" FOREIGN KEY ("idCustom")

REFERENCES public."MuscuEval" ("idCustom") MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

)

WITH (

OIDS = FALSE

)

TABLESPACE pg\_default;

ALTER TABLE public."Record"

OWNER to postgres;

DELETE Script :

DELETE FROM public."Record"

WHERE <condition>;

INSERT Script :

INSERT INTO public."Record"(

"idPerf", "idCustom", "maxBench", "maxSquat", "datePerf")

VALUES (?, ?, ?, ?, ?);

SELECT Script :

SELECT "idPerf", "idCustom", "maxBench", "maxSquat", "datePerf"

FROM public."Record";

UPDATE Script :

UPDATE public."Record"

SET "idPerf"=?, "idCustom"=?, "maxBench"=?, "maxSquat"=?, "datePerf"=?

WHERE <condition>;

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Manuel de Developpement**

***Les choix techniques :***

Le Fichier  est un document de type File (Java Properties File) contenant les informations necessaires à la classe  , afin de permettre la connection à la base de données "Musculation" . Ainsi , nous changerons les informations sur le document.properties, concernant les mots de passes et login si besoin, hors de la classe principale , par mesure de sécurité. Nous trouverons également des exceptions dans la classe ConnectionDbMuscu permettant de comprendre la nature de l'echec de la connection.

**public** **static** **void** **record = fait appel à la Table "Record" de la Base de données "Musculation"**

**public** **static** **void** **ajouter , lister , maj , supp = font appel à la Table "MuscuEval" de la base de données "Musculation"**

La Table "MuscuEval" est modifiable via Java . Ou en dur sur PgAdmin4 .

La Table "Record" sera non modifiable via java afin de limiter les déclarations malveillantes de record . Les informations seront rentrées en dur directement sur la Table SQL "Record" .

Dans la classe main, nous utilisons un SWITCH afin de faciliter l'ajout d'un autre action, anticiper un futur besoin, ou evolution du programme.