PROJET FIL ROUGE — SYSTÈME DE CONSTRUCTION RP (GARRY'S MOD)

Projet B3 Cybersécurité — Efrei Bordeaux

Conception, développement et déploiement d'un addon Garry's Mod professionnel de construction collaborative pour serveur DarkRP, dans un environnement conteneurisé Docker. Ce projet couvre l'ensemble du cycle de vie : infrastructure, développement, tests, sécurité et documentation. addon links : https://steamcommunity.com/sharedfiles/filedetails/?id=3664157203

TABLE DES MATIÈRES

OBJECTIFS DU PROJET

En tant qu'étudiant B3 Cybersécurité, ce Projet Fil Rouge a pour but de démontrer des compétences transversales :

1. **Infrastructure & DevOps** — Conteneurisation d'un serveur de jeu complet (Docker Compose, volumes, networking)
2. **Développement logiciel** — Addon Garry's Mod en GLua avec architecture client/serveur stricte
3. **Base de données** — Intégration MySQL 8.0 via MySQLOO (prepared statements, schéma relationnel)
4. **Sécurité applicative** — Rate limiting, validation d'entrées, injection SQL, gestion des permissions

5. **Documentation technique** — Guides d'installation, d'utilisation, architecture, journal de développement
6. **Résolution de problèmes** — Debug en conditions réelles sur une stack complexe (Docker + Source Engine + Lua)

L'addon développé est un **système de construction collaborative RP** : un joueur Constructeur sélectionne des props, les sauvegarde en blueprint, les place comme fantômes holographiques, puis n'importe quel joueur peut matérialiser ces fantômes avec des caisses de matériaux. Le système intègre également des véhicules simfphys pour le transport logistique des caisses.

INFRASTRUCTURE DOCKER

Pourquoi Docker ?

Le choix de Docker s'impose pour plusieurs raisons : - **Isolation** : le serveur GMod tourne dans un environnement reproductible, sans polluer le VPS - **Reproductibilité** : `docker compose up -d` suffit pour déployer l'ensemble de l'infrastructure - **Snapshots** : `docker commit` permet de sauvegarder l'état du serveur après le téléchargement des ~8 Go de Workshop, évitant de re-télécharger à chaque rebuild - **Séparation des services** : GMod et MySQL dans des containers séparés, communiquant via réseau Docker interne

Architecture Docker

```
VPS Hostinger (16 GB RAM, Ubuntu)
├── Container: gmod-server (ceifa/garrysmod)
│   ├── Port 27015 TCP/UDP
│   ├── Gamemode: DarkRP
│   ├── Map: falaise_lbrp_v1
│   ├── Workshop Collection: 2270926906 (~101 addons, ~8 Go)
│   ├── Volumes bind-mount:
│   │   ├── ./addons → /garrysmod/addons
│   │   ├── ./gamemodes/darkrp → /garrysmod/gamemodes/darkrp
│   │   ├── ./lua-bin → /garrysmod/lua/bin (MySQLOO)
│   │   └── ./server-config/server.cfg → /garrysmod/cfg/server.cfg
│   └── Limite: 3 Go RAM, 2 CPUs
│
├── Container: gmod-mysql (MySQL 8.0)
│   ├── Port 3306
│   ├── Base: gmod_construction
│   ├── Healthcheck: mysqladmin ping
│   └── Limite: 512 Mo RAM, 0.5 CPU
│
└── Volume nommé: gmod-server-data (données persistantes)
```

Choix de l'image

L'image **`ceifa/garrysmod`** a été choisie car c'est la seule image Docker maintenue activement pour les serveurs Garry's Mod. Elle gère automatiquement le téléchargement de SteamCMD et du serveur dédié, ainsi que le Workshop via la variable d'environnement `ARGS`.

docker-compose.yml (résumé)

```
services:
  gmod:
    image: projetfilrouge/gmod-server:v1.1-mysql  # Image commitée après Workshop DL
    container_name: gmod-server
    ports: ["27015:27015/udp", "27015:27015/tcp"]
    mem_limit: 3G
    environment:
      - GAMEMODE=darkrp
      - MAP=falaise_lbrp_v1
      - ARGS=+host_workshop_collection 2270926906 +workshop_download_item 4000 3664157203
    depends_on: [mysql]

  mysql:
    image: mysql:8.0
    container_name: gmod-mysql
    mem_limit: 512M
    environment:
      MYSQL_ROOT_PASSWORD: "***"
      MYSQL_DATABASE: gmod_construction
```

DEUX VERSIONS DE L'ADDON

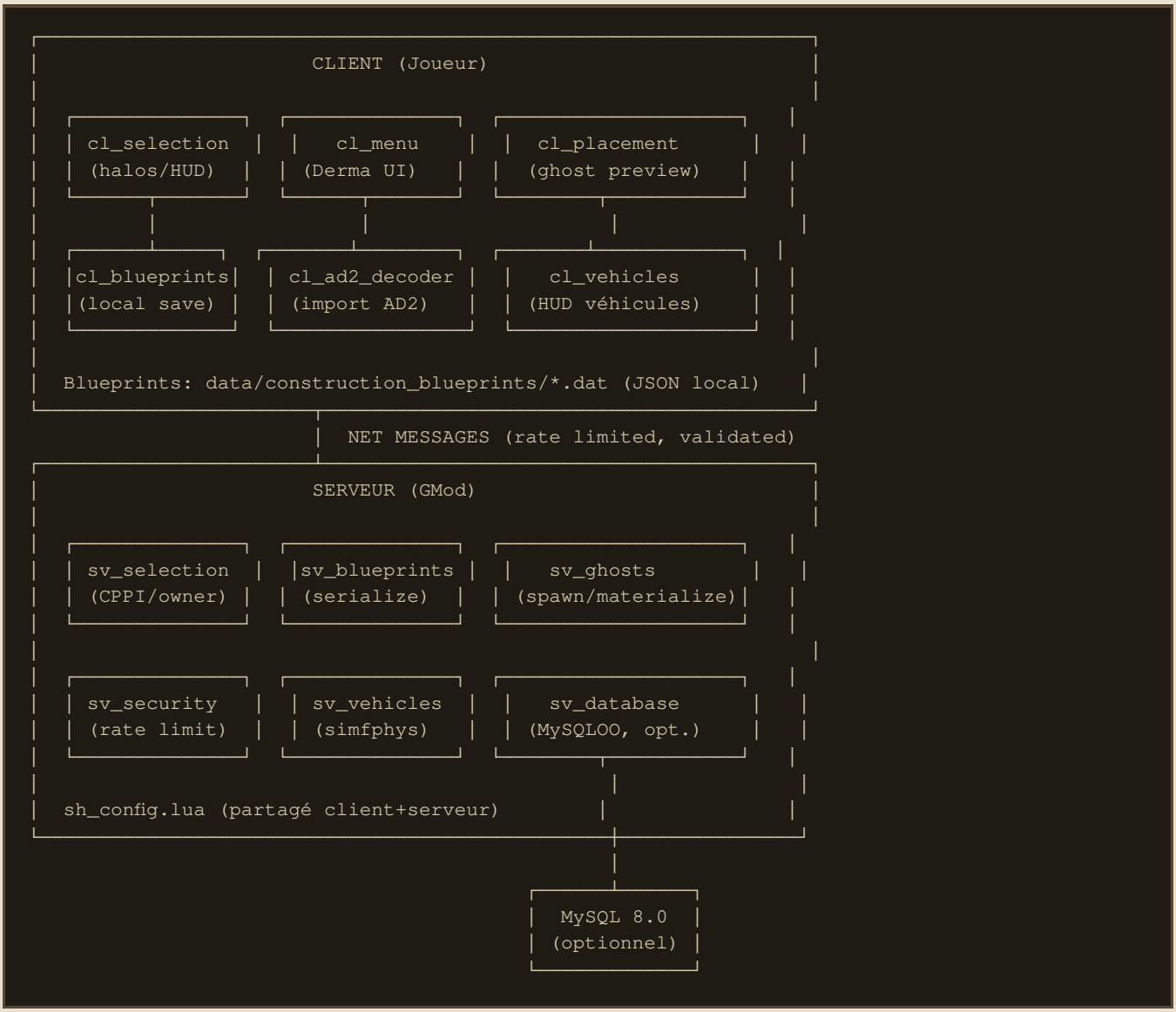
L'addon existe en **deux versions** dans le dossier `addon/`, adaptées à des usages différents :

	DEV `RP_CONSTRUCTION_SYSTEM_DEV/`	**WORKSHOP** `RP_CONSTRUCTION_SYSTEM_WORKSHOP/`
Usage	Serveur Docker avec bind mount	Steam Workshop ou installation manuelle
MySQL	Intégré (MySQLOO, logging DB)	Aucune dépendance externe
Logging	Console + base de données	Console uniquement
Auto-config	`sv_admin_setup.lua` (superadmin)	Absent
Viewmodel	`c_slam.mdl` (fallback dev*)	`v_fortnite_builder.mdl`
Schéma SQL	`sql/schema.sql` fourni	Absent

* En dev Docker avec bind mount, les fichiers custom (modèles) ne sont pas servis aux clients par `resource.AddFile`. Le viewmodel Fortnite Builder n'est disponible côté client qu'en version Workshop.

Le dossier `docker/addons/rp_construction_system/` est une **copie de travail** de la version dev, montée directement dans le container Docker.

ARCHITECTURE TECHNIQUE



Entités custom

`construction_ghost`	Prop fantôme holographique (non-solide, SOLID_NONE), affiché en bleu translucide. Matérialisable par n'importe quel joueur avec une caisse
`construction_crate`	Grosse caisse de matériaux (50 unités), transportable en véhicule simfphys
`construction_crate_small`	Petite caisse (15 unités), utilisable sur place
`weapon_construction`	SWEP dédié : LMB sélection, RMB zone, Shift+RMB menu, R déchargement véhicule/clear

FONCTIONNALITÉS DE L'ADDON V2.2

- **SWEP Construction** — Outil dédié remplaçant le STOOL (plus intuitif, HUD intégré)
- **Blueprints locaux** — Sauvegarde côté client dans `data/construction_blueprints/`, illimitée

- **Dossiers** — Organisation en sous-dossiers comme AdvDupe2
- **Import AdvDupe2** — Décodeur AD2 embarqué, compatible fichiers `.txt`
- **Ghost entities** — Prévisualisation holographique avant construction
- **Construction collaborative** — Tout joueur avec une caisse peut matérialiser les fantômes
- **Caisses de matériaux** — 2 tailles (50 et 15 matériaux), achetables au F4
- **Véhicules simfphys** — Chargement/déchargement de caisses via touche R
- **Offsets calibrés** — Positions de cargo par modèle de véhicule (WW2 Opel, CCKW 6x6, etc.)
- **Net message** `Construction_VehicleReload` — Touche R pour décharger une caisse du véhicule visé
- **Interface moderne** — UI dark theme avec sidebar, navigation par dossiers, badges AD2, breadcrumb
- **Sécurité complète** — Rate limiting, validation serveur, blacklist, vérification des jobs

---

PROBLÈMES RENCONTRÉS ET SOLUTIONS

1. MySQLOO — Binaire 32-bit vs 64-bit

Problème : Après installation de MySQLOO dans `lua/bin/`, le module ne se chargeait pas. Aucune erreur explicite côté serveur, mais `require("mysqloo")` échouait silencieusement.

Cause : Le serveur GMod dans le container Docker `ceifa/garrysmod` tourne en **64-bit**. J'avais téléchargé le binaire `gmsv_mysqloo_linux.dll` (32-bit) au lieu de `gmsv_mysqloo_linux64.dll`.

Solution : Télécharger le binaire 64-bit depuis les releases MySQLOO et le monter via bind mount dans `lua/bin/`. Le nommage est critique : `gmsv_mysqloo_linux64.dll`.

2. Workshop Collection — ~101 addons, ~8 Go

Problème : Chaque `docker compose up -d` depuis une image propre déclenchait le téléchargement complet de la collection Workshop (101 addons, ~8 Go), prenant 5-8 minutes minimum.

Solution : Après le premier démarrage réussi, sauvegarder l'état du container via `docker commit` :

```
docker commit gmod-server projetfilrouge/gmod-server:v1.1-mysql
```

Les démarrages suivants utilisent cette image commitée et sont quasi-instantanés.

3. `docker restart` vs `docker compose up -d` pour les variables d'environnement

Problème : Après modification des variables d'environnement dans `docker-compose.yml` (par ex. changement de map), `docker restart gmod-server` ne prenait pas en compte les changements.

Cause : `docker restart` redémarre le container existant **avec ses anciennes variables**. Seul `docker compose up -d` recrée le container avec les nouvelles valeurs du fichier compose.

Solution : Toujours utiliser `docker compose up -d` après modification du `docker-compose.yml`. Cela s'applique aussi aux changements de map : changer `MAP=falaise_lbrp_v1` nécessite un `docker compose up -d`, pas un simple restart.

4. `resource.AddFile` ne fonctionne pas avec les bind mounts Docker

Problème : Les modèles custom (viewmodel Fortnite Builder) ne se téléchargeaient pas chez les clients. `resource.AddFile()` était appelé côté serveur mais les fichiers n'étaient jamais envoyés.

Cause : En développement avec bind mounts Docker (`./addons:/garrysmod/addons`), le serveur GMod ne sert pas correctement les fichiers custom aux clients. Le système de téléchargement FastDL/resource.AddFile s'attend à ce que les fichiers soient dans le filesystem natif du container, pas dans un volume monté.

Solution : Deux approches selon le contexte : - **En développement** (bind mount) : utiliser un modèle fallback du jeu de base (`c_slam.mdl`) - **En production** (Workshop) : pointer sur le vrai modèle (`v_fortnite_builder.mdl`), téléchargé automatiquement par Steam

La version finale utilise le modèle Workshop. L'addon est publié sur le Steam Workshop (ID 3664157203) avec `+workshop_download_item 4000 3664157203` dans les arguments serveur, ce qui garantit que les clients reçoivent les fichiers custom.

5. `SWEP:Reload()` non appelé côté serveur avec `ClipSize = -1`

Problème : La fonction `SWEP:Reload()` n'était jamais exécutée côté serveur. Le code dans cette fonction ne s'exécutait tout simplement pas quand le joueur appuyait sur R.

Cause : Quand `Primary.ClipSize = -1` (pas de munitions), le moteur Source considère qu'il n'y a rien à recharger et **n'appelle jamais** `Reload()` côté serveur. C'est un comportement documenté mais contre-intuitif du moteur.

Solution : Utiliser un **net message** explicite. `SWEP:Reload()` est défini côté **client uniquement** et envoie un net message `Construction_VehicleReload` au serveur :

```
function SWEP:Reload()
    if SERVER then return end
    net.Start("Construction_VehicleReload")
    net.SendToServer()
end
```

Le serveur reçoit le message et exécute la logique (déchargement véhicule ou clear sélection).

6. `KeyPress` hook ne capte pas `IN_RELOAD`

Problème : Avant d'utiliser le net message, j'avais essayé d'utiliser le hook `KeyPress` côté serveur pour détecter quand le joueur appuie sur R :

```
hook.Add("KeyPress", "Construction_Reload", function(ply, key)
    if key == IN_RELOAD then ... end
end)
```

Mais le callback n'était jamais déclenché pour `IN_RELOAD`.

Cause : Le hook `KeyPress` de GMod ne capture pas tous les IN_ flags. `IN_RELOAD` n'est pas transmis de manière fiable via ce hook, surtout quand une SWEP est active.

Solution : Abandonner `KeyPress` au profit du net message depuis `SWEP:Reload()` côté client (voir problème 5). Pour le HUD véhicule côté client, utiliser `PlayerBindPress` avec `"+reload"` :

```
hook.Add("PlayerBindPress", "Construction_VehicleBind", function(ply, bind, pressed)
    if string.find(bind, "+reload") then ... end
end)
```

7. `SetParent()` crée des ghost physics

Problème : Quand une caisse est attachée à un véhicule simfphys via `SetParent()`, la physique de la caisse reste active et crée une "ghost physics" — le moteur physique continue de simuler l'objet à son ancienne position, causant des collisions invisibles et des comportements erratiques.

Cause : `SetParent()` ne désactive pas automatiquement la physique de l'entité enfant. L'objet physique continue d'exister et de bouger indépendamment du parent.

Solution : Après `SetParent()`, désactiver manuellement le mouvement physique :

```
crate:SetParent(vehicle)
crate:SetLocalPos(offset.pos)
local phys = crate:GetPhysicsObject()
if IsValid(phys) then
    phys:EnableMotion(false)
end
```

Note importante : Ne PAS utiliser `PhysicsDestroy()` — cela causerait des crashs si d'autres systèmes (FPP, DarkRP) tentent d'accéder à l'objet physique. `EnableMotion(false)` suffit.

8. `SetParent(nil)` restore l'ancienne position

Problème : Quand on détache la caisse du véhicule avec `SetParent(nil)`, la caisse se téléporte à sa position **avant** l'attachement, pas à la position actuelle du véhicule.

Cause : Le moteur Source sauvegarde la position locale de l'entité au moment du `SetParent()`. Quand on fait `SetParent(nil)`, il restaure cette position sauvegardée.

Solution : Sauvegarder la position monde **avant** le `SetParent(nil)`, puis téléporter l'entité avec un `timer.Simple(0)` pour attendre que le déparentage soit effectif :

```
local dropPos = crate:GetPos()  -- Position monde actuelle
crate:SetParent(nil)
-- SetPos immédiat ne marche pas car le déparentage est asynchrone
timer.Simple(0, function()
    if IsValid(crate) then
        crate:SetPos(dropPos)
    end
end)
```

En pratique, j'ai calculé une position de drop sur le côté du véhicule pour que la caisse ne tombe pas à travers le véhicule.

9. GMod client cache les fichiers Lua — Reconnexion nécessaire

Problème : Après modification d'un fichier Lua côté serveur (via bind mount), le client ne voyait pas les changements. L'ancien code continuait de s'exécuter.

Cause : Le client Garry's Mod **cache agressivement** les fichiers Lua téléchargés. Même après un `changelevel` ou un map restart, le client utilise ses fichiers en cache.

Solution : Le client doit se **reconnecter complètement** (disconnect + reconnect) pour forcer le re-téléchargement des fichiers modifiés. En développement, c'est contraignant mais inévitable. Astuce : le raccourci `retry` en console accélère le processus.

10. FPP/DarkRP entity ownership issues

Problème : Les entités custom (`construction_ghost`, `construction_crate`) étaient bloquées par FPP (Falco's Prop Protection). Les joueurs ne pouvaient pas interagir avec les caisses des autres, et le physgun était refusé sur les caisses.

Cause : FPP utilise CPPI (`CPPIGetOwner()`) pour vérifier la propriété des entités. Les entités custom n'implémentaient pas l'interface CPPI, donc FPP les considérait comme non-possédées et bloquait toute interaction.

Solution : Implémenter des hooks spécifiques pour chaque type d'interaction :

```
hook.Add("PhysgunPickup", "Construction_CratePhysgun", function(ply, ent)
    if ent:GetClass() ~= "construction_crate" then return end
    if ent:GetNWBool("IsLoaded", false) then return false end  -- Pas de physgun si chargée
    local owner = ent:CPPIGetOwner()
    if IsValid(owner) and owner == ply then return true end
    if ply:IsAdmin() then return true end
end)
```

Même approche pour `CanTool`, `GravGunPickupAllowed`, etc.

11. Map change nécessite `docker compose up -d`

Problème : Changer la map dans le `docker-compose.yml` (variable `MAP`) puis faire `docker restart` ne changeait pas la map.

Cause : Identique au problème 3 — `docker restart` ne relit pas le fichier compose. Les variables d'environnement du container restent celles de sa création.

Solution : Toujours utiliser `docker compose up -d` pour appliquer les modifications du fichier compose. Pour un simple changement de map sans modifier le compose, utiliser `changelevel` en RCON.

CHRONOLOGIE DU DÉVELOPPEMENT

Étape 1 — Infrastructure Docker & Structure

- Installation Docker sur VPS Hostinger (16 Go RAM)
- Recherche et test de l'image `ceifa/garrysmod`
- Création du `docker-compose.yml` (GMod + MySQL 8.0)
- Premier démarrage, téléchargement Workshop (~8 Go)
- `docker commit` pour sauvegarder l'image avec le Workshop
- Création de la structure du projet Git

Étape 2 — Configuration DarkRP & MySQL

- Installation de DarkRP (gamemode + darkrpmodification)
- Configuration des jobs (TEAM_BUILDER = Constructeur)
- Installation MySQLOO 64-bit (résolution du bug 32-bit)
- Création du schéma SQL (`blueprint_logs`, `shared_blueprints`)
- Test de connexion MySQLOO via hostname Docker `gmod-mysql`
- Premier commit avec infrastructure fonctionnelle

Étape 3 — Système de sélection (STOOL initial)

- Développement du STOOL `construction_select` (Tool Gun)
- Système de sélection par clic (CPPI ownership check)
- Sélection par rayon (clic droit)
- Rendu visuel avec halos bleus (client)
- Synchronisation client/serveur de la sélection

Étape 4 — Sérialisation & Blueprints

- Sérialisation des props (positions relatives, modèles, physique)
- Résolution : Vector/Angle → table en JSON → reconstruction custom
- Compression : `util.TableToJSON()` → `util.Compress()` → `util.Base64Encode()`
- Sauvegarde/chargement depuis MySQL (prepared statements)
- Batch spawning (5 props par tick, anti-lag)

Étape 5 — Permissions & Partage

- Système de permissions (view, use, edit)
- Partage entre joueurs via SteamID
- Interface Derma pour la gestion des permissions
- Vérification des permissions à chaque action

Étape 6 — Sécurité

- Rate limiting par action (cooldowns configurables)
- Blacklist de classes d'entités (money_printer, drug_lab, etc.)
- Validation serveur de chaque blueprint reçu
- Restrictions par job DarkRP

- Logging de toutes les actions en base de données

Étape 7 — Refonte v2.0 : SWEP + Ghosts + Caisses

- Migration du STOOL vers un **SWEP dédié** (`weapon_construction`)
- Développement des ghost entities (non-solides, holographiques)
- Développement des caisses de matériaux (grosse 50 + petite 15)
- Système de matérialisation (Use sur ghost avec caisse active)
- Résolution des problèmes FPP/CPPI

Étape 8 — Placement avancé & Interface

- Panneau de placement AdvDupe2-style (rotation, hauteur, position originale)
- Prévisualisation holographique avant confirmation
- Décodeur AdvDupe2 embarqué (import fichiers .txt)
- UI moderne dark theme avec sidebar et navigation dossiers

Étape 9 — Sauvegardes locales & Documentation

- Migration des blueprints vers stockage **local client** (`data/construction_blueprints/`)
- Support des sous-dossiers
- Badges AD2 dans l'interface
- Documentation complète (README, ARCHITECTURE, guides)

Étape 10 — Véhicules simfphys v2.2

- Module véhicules serveur (`sv_vehicles.lua`) et client (`cl_vehicles.lua`)
- Détection automatique simfphys, LVS, Source vehicles
- Système d'attachement via `SetParent()` + physique désactivée
- Offsets calibrés par modèle de véhicule WW2
- Résolution des bugs `SetParent()` (ghost physics, position restore)
- Net message `Construction_VehicleReload` (touche R = déchargement)
- HUD véhicule client (instructions charger/décharger)
- `PlayerBindPress` pour la touche R côté client

Étape 11 — Finalisation & Tests

- Tests complets en conditions réelles (multi-joueurs)
- Résolution du problème SWEP:Reload() avec ClipSize=-1
- Résolution du problème KeyPress + IN_RELOAD
- Nettoyage du code, commentaires
- Mise à jour complète de la documentation v2.2

Étape 12 — Publication Workshop & Packaging

- Création du fichier `addon.json` requis par gmad
- Suppression des fichiers `.sw.vtx` (non supportés par la whitelist gmad)
- Compilation du `.gma` via `gmad.exe create`
- Publication sur le Steam Workshop via `gmpublish.exe` (ID 3664157203)
- Création de l'icône 512x512 pour la page Workshop

- Séparation en deux versions : **dev** (MySQL, logging DB) et **workshop** (standalone)
- Basculement du viewmodel serveur vers `v_fortnite_builder.mdl` (fonctionnel via Workshop)
- Ajout de `+workshop_download_item 4000 3664157203` dans les arguments Docker

SÉCURITÉ

La sécurité est un axe majeur de ce projet, cohérent avec la spécialisation B3 Cybersécurité :

Rate Limiting

- Cooldown par action : sauvegarde (10s), chargement (15s)
- Protection contre le spam de net messages
- Cooldown sur les actions véhicule (1s)

Validation des entrées

- Longueur des noms/descriptions (50/200 caractères max)
- Nombre de props par blueprint (150 max configurable)
- Rayon de sélection borné (50–1000 unités)
- Vérification que seuls les `prop_physics` sont autorisés

Blacklist de classes

```
Config.BlacklistedEntities = {
    "money_printer", "darkrp_money", "spawned_money",
    "spawned_shipment", "spawned_weapon",
    "drug_lab", "gun_lab", "microwave", "bitminers_"
}
```

Restrictions par job

- Seuls les jobs configurés peuvent utiliser le SWEP
- Les caisses peuvent être restreintes à certains jobs
- Configuration via `sh_config.lua`

SQL Injection

- Toutes les requêtes utilisent des **prepared statements** via MySQLOO
- Aucune concaténation de strings dans les requêtes SQL
- Échappement automatique des paramètres

Séparation client/serveur

- Architecture stricte : le client n'a jamais confiance côté serveur
- Chaque action client est re-validée côté serveur
- Les blueprints envoyés par le client sont entièrement re-validés (props, classes, limites)

STRUCTURE DU PROJET

```
ProjetFilRouge/
├── addon/                                #   Addon en deux versions
│   ├── rp_construction_system_dev/       #    Version développement (MySQL + logging DB)
│   │   ├── lua/                          #    Code source complet + sv_database.lua
│   │   ├── sql/schema.sql                #    Schéma MySQL
│   │   └── README.md                     #    Doc version dev
│   └── rp_construction_system_workshop/ #    Version Workshop (standalone, sans MySQL)
│       ├── lua/                          #    Code source allégé
│       └── README.md                     #    Doc version workshop
├── docker/                             #   Environnement Docker
│   ├── docker-compose.yml              #    Orchestration GMod + MySQL
│   ├── addons/                         #    Addons montés (copie de dev)
│   │   ├── rp_construction_system/     #    Addon (sync via rsync)
│   │   └── darkrpmodification/       #    Config DarkRP (jobs, entities)
│   ├── lua-bin/                        #    MySQLOO binaires 64-bit
│   ├── mysql-init/                     #    Script init SQL
│   └── server-config/                #    server.cfg
├── docs/                               #   Documentation technique
│   ├── ARCHITECTURE.md                 #    Architecture détaillée
│   ├── GUIDE_INSTALLATION.md           #    Guide admin serveur
│   ├── GUIDE_UTILISATEUR.md            #    Guide joueur
│   ├── DOCKER_IMAGES.md                #    Gestion images Docker
│   └── JOURNAL_DEV.md                  #    Journal de développement
├── rendu/                              #   Rendus académiques (Projet Fil Rouge)
│   ├── dat/                            #    Dossier d'Architecture Technique
│   ├── amelioration/                   #    Compte-rendu d'amélioration
│   ├── backup/                         #    Plan de sauvegarde
│   └── cdc/                            #    Cahier des charges
├── screenshots/                        #   Icône et captures
│   ├── icon_512x512.png                #    Icône addon (PNG)
│   └── icon_512x512.jpg                #    Icône addon (JPG Workshop)
└── README.md                           #    Ce fichier
```

STACK TECHNIQUE

VPS	Hostinger	16 Go RAM, Ubuntu
Serveur de jeu	Garry's Mod	Docker: `ceifa/garrysmod`
Gamemode	DarkRP	Avec `darkrpmodification`
Base de données	MySQL 8.0	Container Docker, optionnel
Module DB	MySQLOO 9.7	Binaire 64-bit Linux
Langage	GLua	Garry's Mod Lua (basé sur Lua 5.1)
Versioning	Git + GitHub	Repository public
Orchestration	Docker Compose	2 services + volumes
Map	falaise_lbrp_v1	Workshop ID 3174802588
Workshop	Collection 2270926906	~101 addons
Véhicules	simfphys	Support LVS documenté
Addon Workshop	Steam Workshop	ID 3664157203

DOCUMENTATION

README Addon Dev	Documentation version développement (MySQL intégré)
README Addon Workshop	Documentation version Workshop (standalone)
Architecture	Architecture technique détaillée, flux de données, net messages
Guide d'installation	Guide admin serveur (Docker, DarkRP, configuration)
Guide d'utilisation	Guide joueur (contrôles, blueprints, caisses, véhicules)
Images Docker	Gestion des snapshots Docker

CAPTURES D'ÉCRAN

RP Construction System

> **Steam Workshop** — *Addon publié (ID 3664157203)*

CONCLUSION ET PERSPECTIVES

Ce Projet Fil Rouge m'a permis de couvrir un spectre technique large, de l'infrastructure Docker à la programmation réseau Lua, en passant par la gestion de base de données et la sécurité applicative. Chaque problème rencontré a été une occasion d'approfondir ma compréhension du moteur Source, du protocole réseau de GMod, et des subtilités de Docker.

Réalisations

- Infrastructure Docker complète et reproductible
- Addon v2.2 publié sur le Steam Workshop
- Architecture client/serveur sécurisée
- Système de construction collaborative fonctionnel
- Intégration véhicules simfphys
- Documentation technique exhaustive
- Deux versions : dev (MySQL + logging) et workshop (standalone)

Perspectives

- Système de coûts configurable (matériaux par prop)
- Support étendu des véhicules LVS
- Système de partage de blueprints entre joueurs