Product Display Application

Last Modification Date: 10/06/2025

Table of Contents

1. Introduction

- Purpose: Product display with 5 different modes of display and control panel, the content to display will be according to the control panel settings where only the administrator user can access. For security reasons, the application only will be accessible through a HTTPs reverse proxy. Additionally the displays where the application will be displayed will turn on at 8:00 am and turn off at 17:00 pm automatically.
- Scope of the documentation: Hardware and software requirements, diagrams of the application and the database, structure of the application and installation & deployment
- Audience: Store customers

2. Requirements

Hardware	
CPU	>= 1 GHz
RAM	>= 8 GB
HDD	>= 50 GB

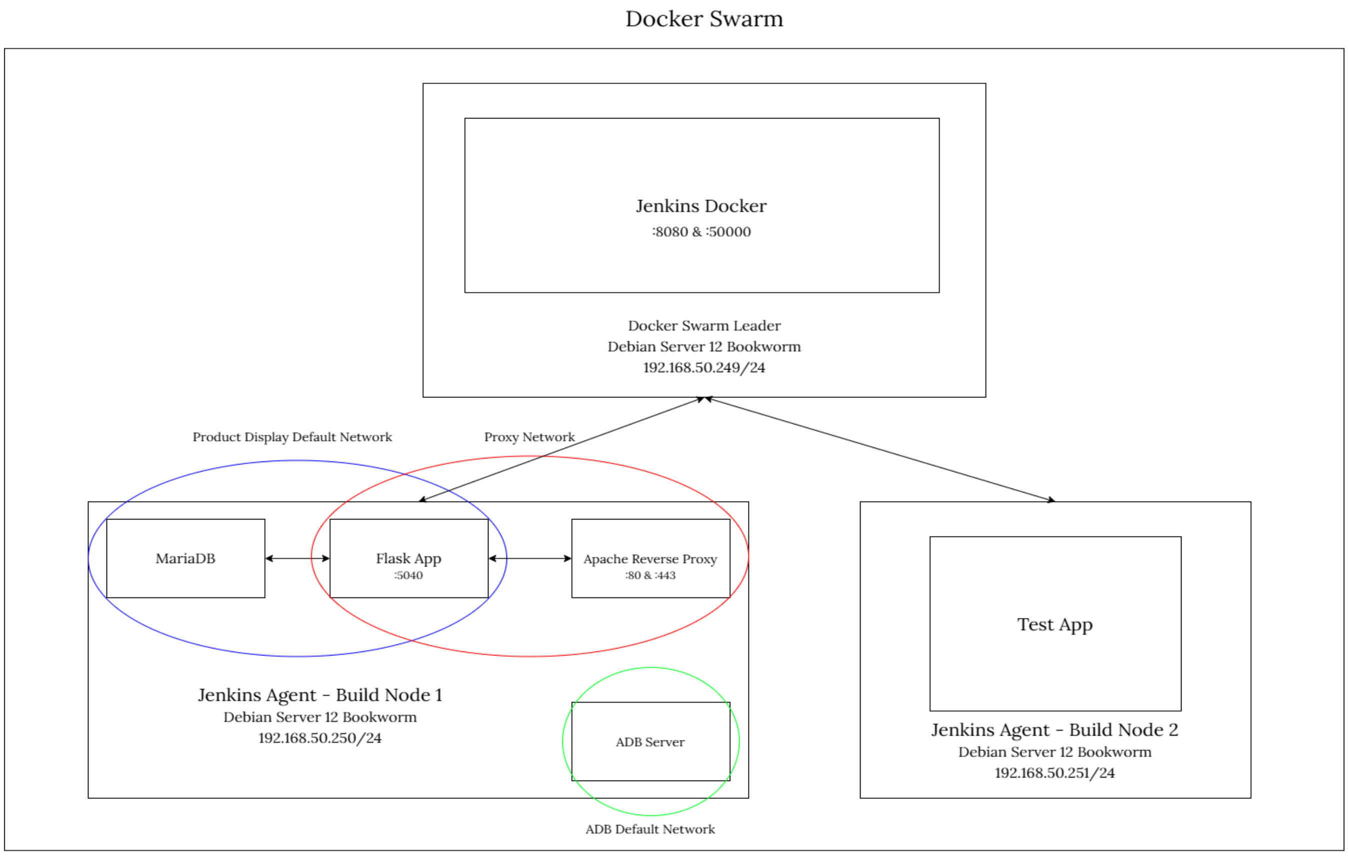
Software	
Docker Engine	20.10.24
Docker Compose	1.29.2
Jenkins	2.504.1
MariaDB	11.7.2
Python	3.13
Git	2.49.0
Proxmox	8.4
Apache	2.4.63 (Unix)
Flask	3.1.1

Operating Systems	
Debian (Linux GNU)	12 Bookworm

3. General Architecture

3.1 Diagram of Components

- Jenkins docker container
- Jenkins agent (build node 1)
  - Apache reverse proxy docker container: Allows the access to the application through the ports 80 and 443
  - MariaDB docker container: Stores the database in docker volume
  - Flask app docker container: Runs the application
  - ADB docker container: Turns on and off the displays of the shop (8:00 am to 17:00 pm) and executes the "Kiosk Fully Browser" app
- Jenkins agent (build node 2)
  - Test environment: For testing purposes



3.2 Deployment (Docker Compose)

- Services:
  1. Jenkins
  2. MariaDB
  3. App: Depends on MariaDB
  4. Apache
  5. ADB Server

- Networks:
  - product-display_default ⇒ MariaDB & App
  - proxy_network ⇒ Apache & App
  - adb-docker_default ⇒ ADB Server
  - jenkins-docker_default ⇒ Jenkins

- Volumes:
  - database ⇒ Stores the MariaDB database
  - screenshots ⇒ Stores screenshots made by the application with playwright
  - adb_keys ⇒ Stores fingerprints of the ADB Server
  - jenkins-data ⇒ Stores all the information of jenkins

3.3 Structure of the Database

3.3.1 Creation Script (init.sql)

```sql
CREATE TABLE categories (
    category_id INT PRIMARY KEY,
    category_name VARCHAR(256) NOT NULL,
    category_image VARCHAR(256) NULL
);
CREATE TABLE products (
    product_id INT PRIMARY KEY,
    product_name VARCHAR(128) NOT NULL,
    product_description TEXT,
    product_sku VARCHAR(64),
    product_image VARCHAR(256) NOT NULL,
    product_price VARCHAR(32) NOT NULL,
    product_modification DATETIME NOT NULL,
    product_category INT,
    FOREIGN KEY (product_category) REFERENCES categories(category_id)
);
CREATE TABLE product_categories (
    product_id INT,
    category_id INT,
    PRIMARY KEY (product_id, category_id),
    FOREIGN KEY (product_id) REFERENCES products(product_id),
    FOREIGN KEY (category_id) REFERENCES categories(category_id)
);
```

```sql
CREATE TABLE probabilities (
    id INT PRIMARY KEY AUTO_INCREMENT,
    random_product INT NOT NULL,
    random_category_products INT NOT NULL,
    grid INT NOT NULL,
    youtube_video INT NOT NULL,
    website INT NOT NULL
);

CREATE TABLE videos (
    id VARCHAR(16) PRIMARY KEY,
    video_url VARCHAR(256) NOT NULL,
    video_title VARCHAR(256) NOT NULL,
    video_length INT NOT NULL
);

CREATE TABLE websites (
    id INT PRIMARY KEY AUTO_INCREMENT,
    website_url VARCHAR(256) NOT NULL
);

CREATE TABLE blacklisted (
    id INT PRIMARY KEY AUTO_INCREMENT,
    type ENUM('product', 'category') NOT NULL,
    name VARCHAR(256) NOT NULL
);

CREATE TABLE speed (
    id INT PRIMARY KEY AUTO_INCREMENT,
    speed_ms INT NOT NULL
);
```

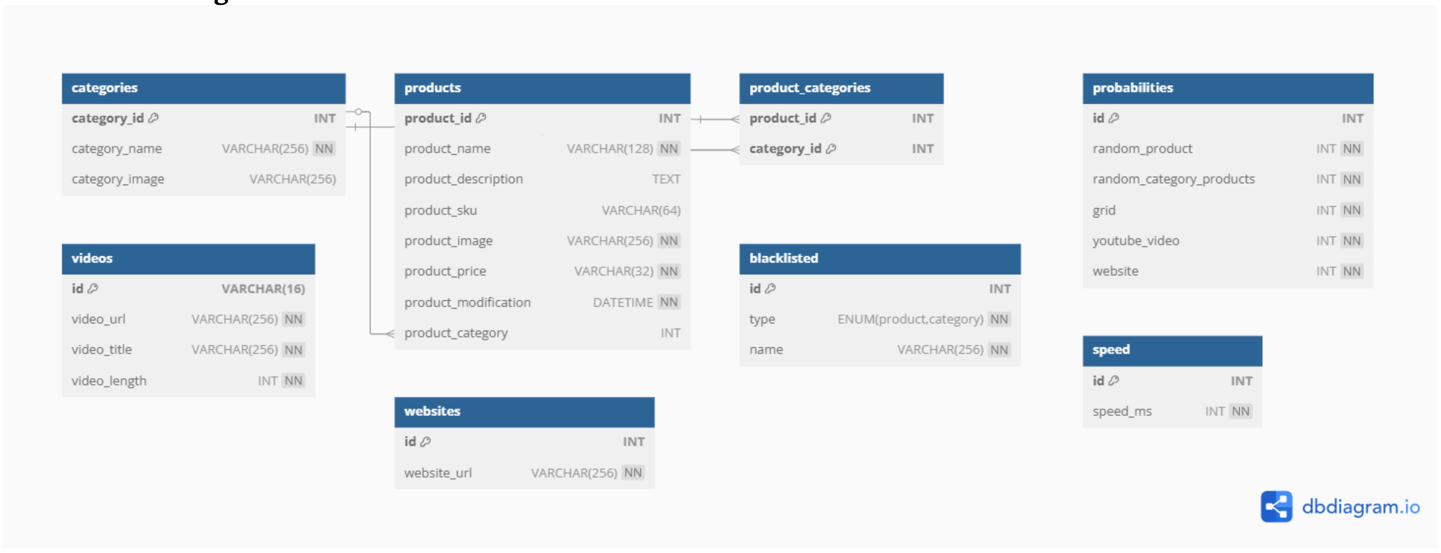
3.3.2 Indexes

- category_id in categories (primary key)
- product_id in products (primary key)
- product_id and category_id in product_categories (composite primary key)
- Foreign key indexes for product_category in products and both keys in product_categories

3.3.3 Tables

- products: Contains the information of all the products in the woocommerce shop
- product_categories: Contains the relation between products and their categories
- categories: Contains the information of all categories in the woocommerce shop
- blacklisted: Contains blacklisted products and categories (type)
- probabilities: Contains the probabilities of the modes
- speed: Contains the speed of the random product mode
- videos: Contains the information of the youtube videos
- websites: Contains the information of the websites. The ID is related with the screenshot name

3.3.4 Diagram of the database



3.3.5 Backup, restoring and transferring

The following package must be installed: mysql-client

Backup

```
mysqldump -u root -p --opt --column-statistics=0 db > backup.sql
```

Restoring database

```
mysql -u root -p -e "CREATE DATABASE db;"
mysql -u root -p db < backup.sql
```

Transferring the database to a new server

```
scp backup.sql user@server:/path
```

4. Structure and operation of the flask app

4.1 Structure

```
product-display/
├─ app/
│  ├─ html/
│  │  ├─ control_panel.html
│  │  ├─ index.html/
│  ├─ static/
│  │  ├─ css/
│  │  │  ├─ index.css
│  │  │  ├─ control_panel.css
│  │  ├─ js/
│  │  │  ├─ index.js
│  │  │  ├─ control_panel.js/
│  │  ├─ img/
│  ├─ app.py
│  ├─ Dockerfile
│  ├─ requirements.txt
├─ mariadb/
│  ├─ Dockerfile
│  ├─ init.sql
├─ docker-compose.yml
├─ Jenkinsfile
```

4.2 Description of the folders

- product-display: Contains the application and the mariadb
- app: Contains the application build files
  - html: Contains the html template files for the flask app
  - static
    - css: Contains the index.html and control_panel.html styles
    - js: Contains the javascript for the index.html and control_panel.html
    - img: Contains the screenshots made by the application. This folder is mounted in a docker volume
- mariadb: Contains the mariadb build files

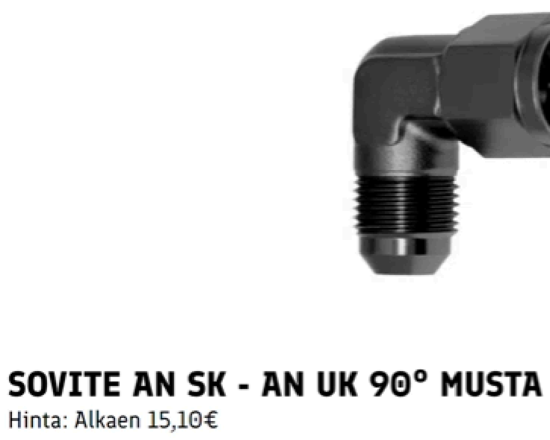
4.3 Application operation and display modes

The application has five different display modes that are executed depending on what the main function returns (depends on the chances set in the control panel). These modes are:

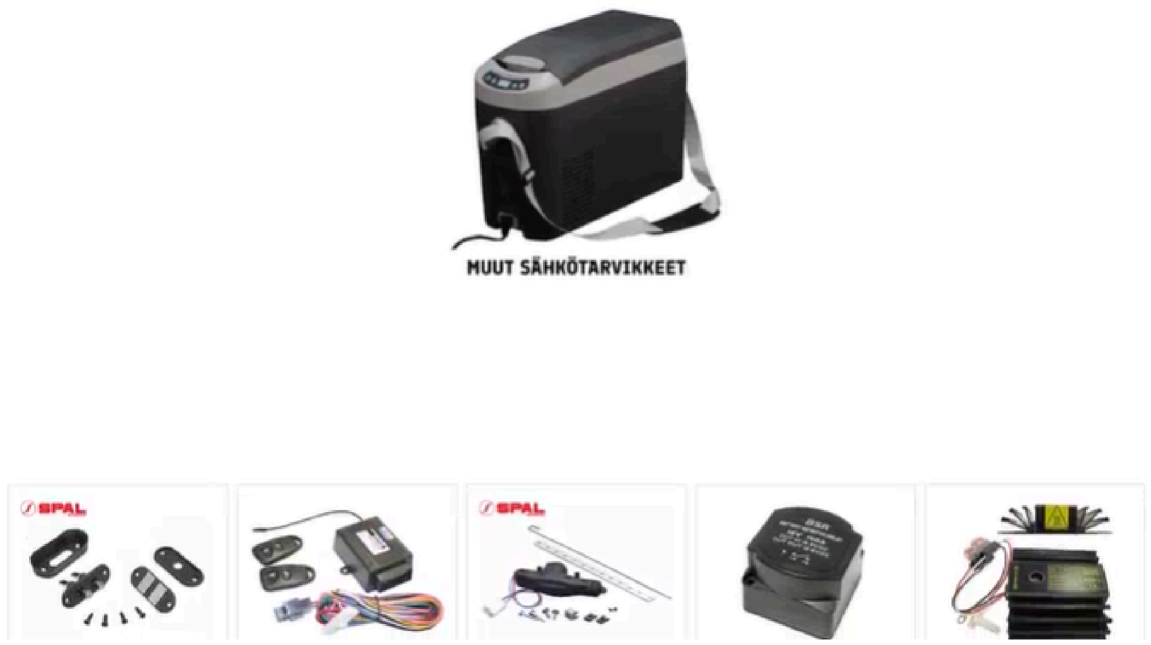
- Random product: Displays a random product from the database for an amount of time (set in the control panel) *[Image 1]*

- Random category products: Displays the products of a random category/subcategory in a auto scrollable grid *[Image 2]*
- Grid of random products: Displays an auto scrollable grid of 198 random products *[Image 3]*
- Youtube video: Displays a random youtube video from the database (set in the control panel) *[Image 4]*
- Random website: Displays the screenshot of a random website (set in the control panel). *"Since iframe was getting rejected I think that this was the most suitable option" [Image 5]*

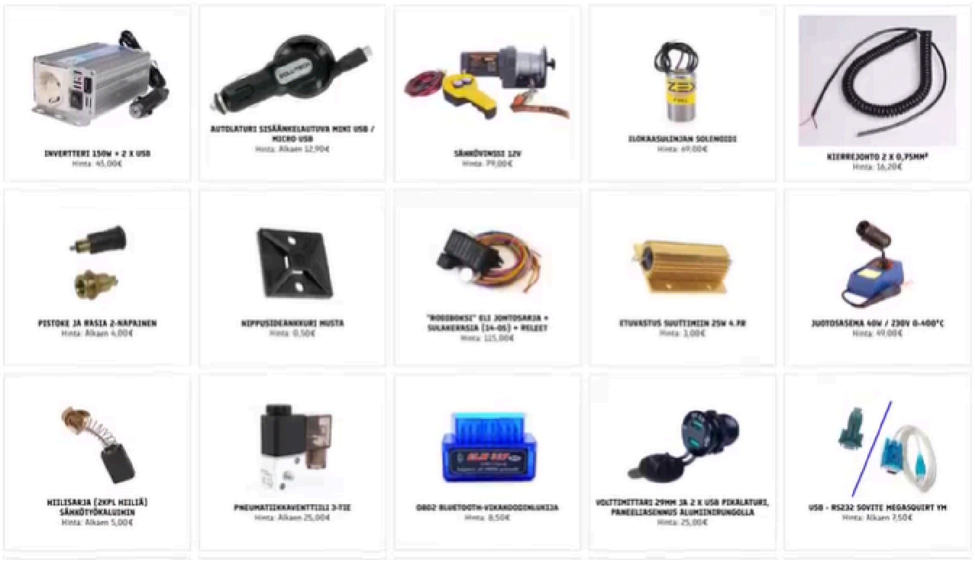
➢ [Image 1]


SOVITE AN SK - AN UK 90° MUSTA
Hinta: Alkaen 15,10€

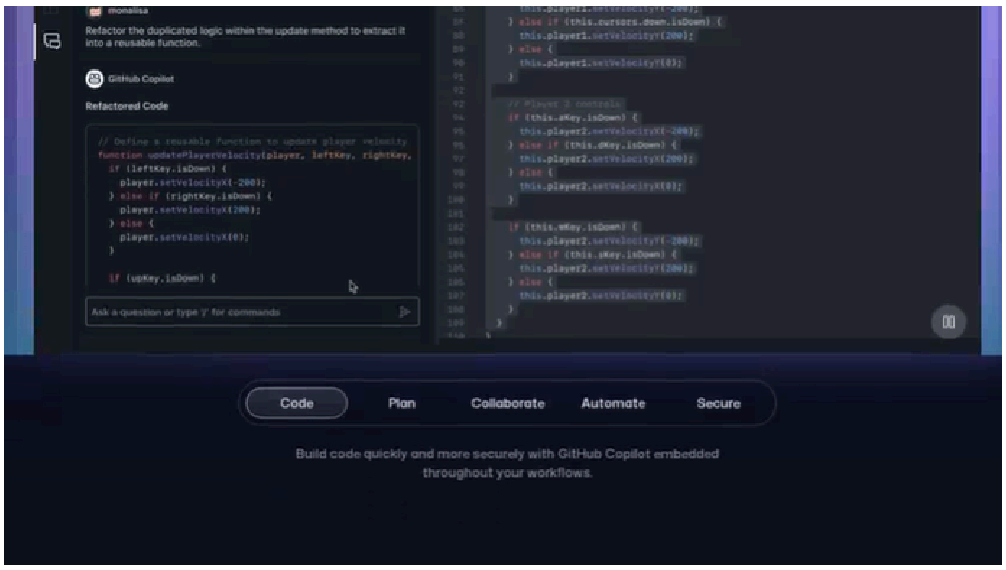
➢ [Image 2]



➢ [Image 3]


INVERTTERI 150W + 2 X USB
Hinta: 45,00€
AUTOLATURI SISÄÄNKELAUTUVA MINI USB / MICRO USB
Hinta: Alkaen 12,90€
SÄHKÖVINSSI 12V
Hinta: 79,00€
ILOKAASULINJAN SOLENOIDI
Hinta: 69,00€
KIERREJOHTO 2 X 0,75MM²
Hinta: 16,20€
PISTOKE JA RASIA 2-NAPAINEN
Hinta: Alkaen 4,00€
NIPPUSIDEANKKURI MUSTA
Hinta: 0,50€
"ROODBOKSI" ELI JOHTOSARJA + SULAKERASIA (14-05) + RELEET
Hinta: 115,00€
ETUVASTUS SUUTTIMIIN 25W 4 PR
Hinta: 3,00€
JUOTOSASEMA 40W / 230V 0-400°C
Hinta: 49,00€
HIILISARJA (2KPL HIILIÄ) SÄHKÖTYÖKALUIHIN
Hinta: Alkaen 5,00€
PNEUMATIIKKAVENTTIILI 3-TIE
Hinta: Alkaen 25,00€
OBD2 BLUETOOTH-VIKAKOODINLUKIJA
Hinta: 8,50€
VOLTTIMITTARI 29MM JA 2 X USB PIKALATURI, PANEELIASENNUS ALUMIINIRUNGOLLA
Hinta: 25,00€
USB - RS232 SOVITE MEGASQUIRT YM
Hinta: Alkaen 7,50€

➢ [Image 4]


monalisa

Refactor the duplicated logic within the update method to extract it into a reusable function.

GitHub Copilot

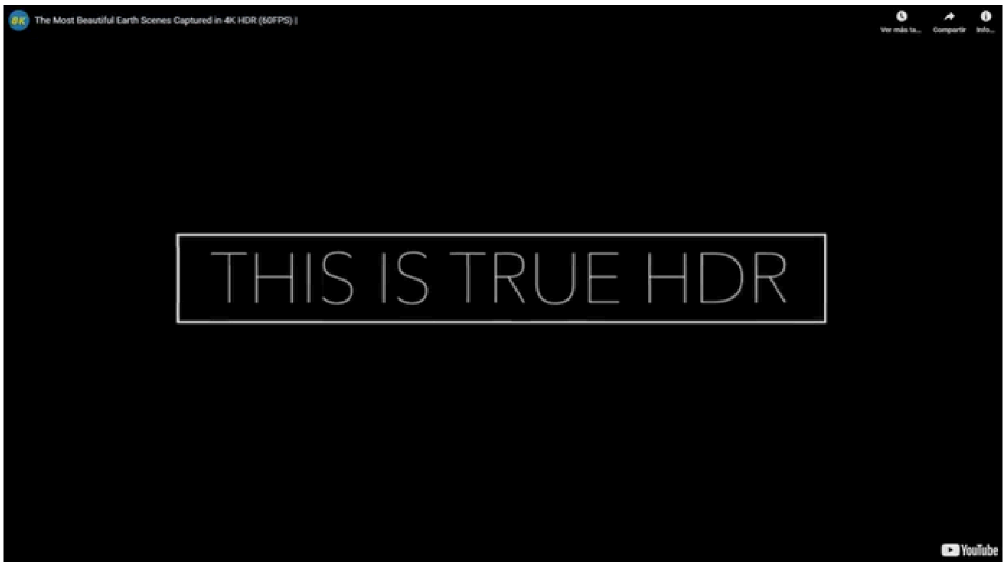
Refactored Code

```
// Define a reusable function to update player velocity
function updatePlayerVelocity(player, leftKey, rightKey,
  if (leftKey.isDown) {
    player.setVelocityX(-200);
  } else if (rightKey.isDown) {
    player.setVelocityX(200);
  } else {
    player.setVelocityX(0);
  }

  if (upKey.isDown) {
```

```
    this.player1.setVelocityY(-200);
  } else if (this.cursors.down.isDown) {
    this.player1.setVelocityY(200);
  } else {
    this.player1.setVelocityY(0);
  }

  // Player 2 controls
  if (this.aKey.isDown) {
    this.player2.setVelocityX(-200);
  } else if (this.dKey.isDown) {
    this.player2.setVelocityX(200);
  } else {
    this.player2.setVelocityX(0);
  }

  if (this.wKey.isDown) {
    this.player2.setVelocityY(-200);
  } else if (this.sKey.isDown) {
    this.player2.setVelocityY(200);
  } else {
    this.player2.setVelocityY(0);
  }
 }
}
```

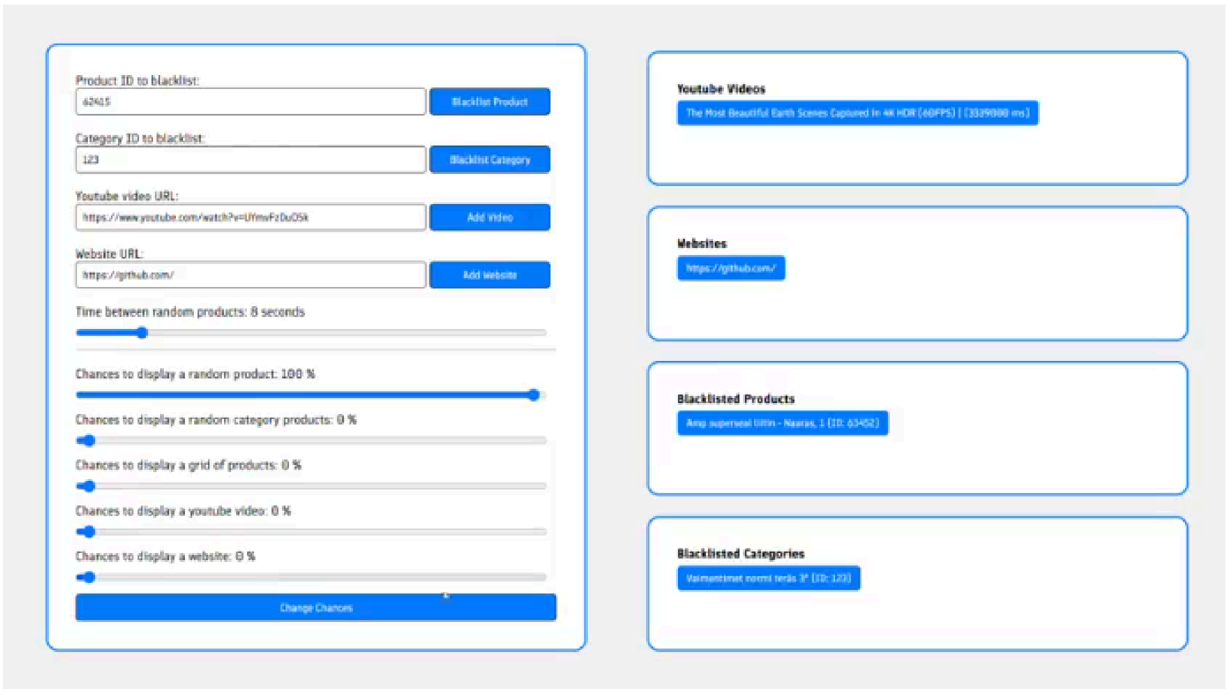
Build code quickly and more securely with GitHub Copilot embedded throughout your workflows.

➢ [Image 5]


The Most Beautiful Earth Scenes Captured in 4K HDR (60FPS) |
Ver más ta...
Compartir
Info...
THIS IS TRUE HDR
YouTube

4.4 Application control panel

The application control panel can be accessed through the following route: /admin. There you can control parameters like blacklisted products, blacklisted categories, youtube videos and websites to display, seconds between random products and chances between modes



Product ID to blacklist: 62415

Category ID to blacklist: 123

Youtube video URL: https://www.youtube.com/watch?v=UfmvFzDuO5k

Website URL: https://github.com/

Time between random products: 8 seconds

Chances to display a random product: 100 %

Chances to display a random category products: 0 %

Chances to display a grid of products: 0 %

Chances to display a youtube video: 0 %

Chances to display a website: 0 %

Youtube Videos

The Most Beautiful Earth Scenes Captured In 4K HDR (60FPS) | (3339000 ms)

Websites

https://github.com/

Blacklisted Products

Amp superseal Uitin - Naaras, 1 (ID: 63452)

Blacklisted Categories

Vaimentimet normi teräs 3" (ID: 123)

5. Disaster recovery plan (replicate)

5.1 Impact analysis

Component	Impact IF Fails	Criticity Level
MariaDB	Loss of Products	High (Not critic since the database can be rebuilt from WooCommerce by executing the check_updates() function in app.py)
Flask App	Product Display Not Available	High
Apache Reverse Proxy	Web Access Not Available	High
ADB Server	Displays Do Not Turn On/Off Automatically and The Product Display App is Not Opened	Medium
Jenkins	Automatization and Deploys Interruption	Low

5.2 Recovery strategy

For the database as mentioned before we can backup, restore and transfer to another server using the following method.

For the critical docker volumes, we can backup them using the following commands:

```
docker run --rm -v product-display_screenshots:/volume -v
$(pwd):/backup alpine \
tar czf /backup/screenshots_backup.tar.gz -C /volume .

docker run --rm -v product-display_database:/volume -v $(pwd):/backup
alpine \
tar czf /backup/database_backup.tar.gz -C /volume .

docker run --rm -v adb-docker_adb_keys:/volume -v $(pwd):/backup alpine
\
tar czf /backup/adb_keys_backup.tar.gz -C /volume .

docker run --rm -v jenkins-docker_jenkins-data:/volume -v
$(pwd):/backup alpine \
tar czf /backup/jenkins_data_backup.tar.gz -C /volume .
```

And restore them using the following commands:

```
docker volume create product-display_screenshots
docker volume create product-display_database
docker volume create adb-docker_adb_keys
docker volume create jenkins-docker_jenkins-data

docker run --rm -v product-display_screenshots:/volume -v
$(pwd):/backup alpine \
tar xzf /backup/screenshots_backup.tar.gz -C /volume

docker run --rm -v product-display_database:/volume -v $(pwd):/backup
alpine \
tar xzf /backup/database_backup.tar.gz -C /volume

docker run --rm -v adb-docker_adb_keys:/volume -v $(pwd):/backup alpine
\
tar xzf /backup/adb_keys_backup.tar.gz -C /volume

docker run --rm -v jenkins-docker_jenkins-data:/volume -v
$(pwd):/backup alpine \
tar xzf /backup/jenkins_data_backup.tar.gz -C /volume
```

5.2.1 Recovery Procedure Step-To-Step

1. Set up the new physical or cloud server
   1.1. Create required virtual machines with Debian 12 Bookworm
   1.2. Assign static IPs (if needed) and proper hostnames
2. Install docker engine and docker compose in each virtual machine

```
apt update

apt install docker.io docker-compose -y
```

3. Create and join them in the same docker swarm

```
docker swarm init

docker swarm join --token <TOKEN>
```

4. Restore the volumes
   4.1. Transfer .tar.gz backups to the corresponding VM
   4.2. Execute the following commands for each volume

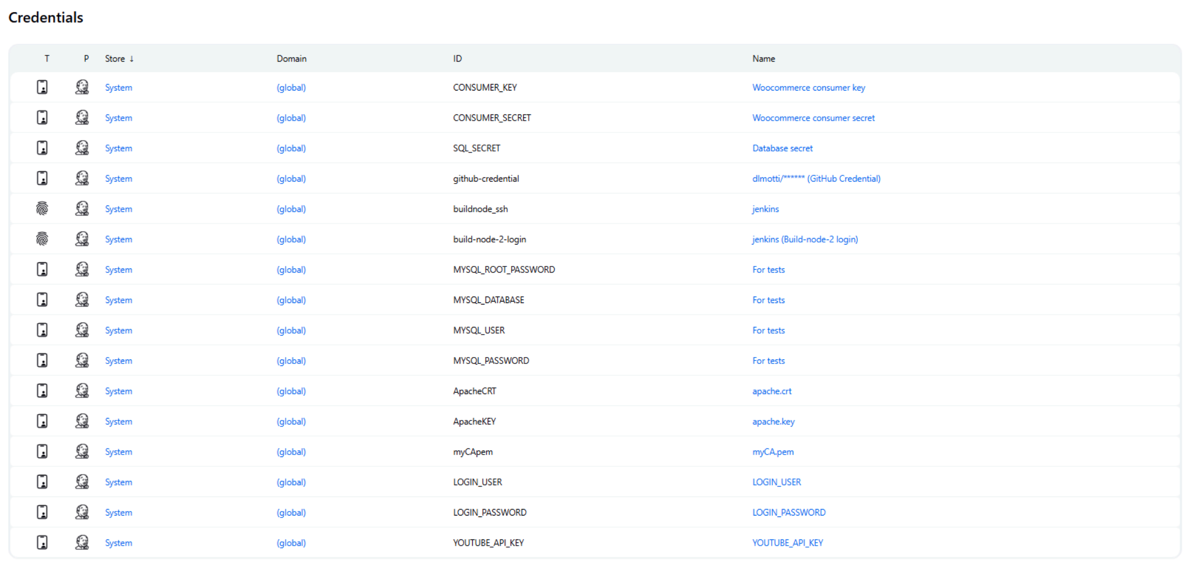
```
docker volume create <volume-name>

docker run --rm -v <volume-name>:/volume -v $(pwd):/backup
alpine \ tar xzf /backup/<volume-name>_backup.tar.gz -C
/volume
```

5. Deploy jenkins

```
docker-compose up -d jenkins-docker
```

6. Deploy all the applications through jenkins
7. Validations
   7.1. Jenkins web user interface is reachable at port 8080
   7.2. App is reachable and functional
   7.3. Control panel is reachable and functional
   7.4. Screenshots are displayed
   7.5. Displays turn on/off via ADB

Note: In case you cannot restore the Jenkins volume you will have to reconfigure and add the credentials to it again



Credentials

T	P	Store ↓	Domain	ID	Name
▯	●	System	(global)	CONSUMER_KEY	Woocommerce consumer key
▯	●	System	(global)	CONSUMER_SECRET	Woocommerce consumer secret
▯	●	System	(global)	SQL_SECRET	Database secret
▯	●	System	(global)	github-credential	dlmotti/****** (GitHub Credential)
◎	●	System	(global)	buildnode_ssh	jenkins
◎	●	System	(global)	build-node-2-login	jenkins (Build-node-2 login)
▯	●	System	(global)	MYSQL_ROOT_PASSWORD	For tests
▯	●	System	(global)	MYSQL_DATABASE	For tests
▯	●	System	(global)	MYSQL_USER	For tests
▯	●	System	(global)	MYSQL_PASSWORD	For tests
▯	●	System	(global)	ApacheCRT	apache.crt
▯	●	System	(global)	ApacheKEY	apache.key
▯	●	System	(global)	myCApem	myCA.pem
▯	●	System	(global)	LOGIN_USER	LOGIN_USER
▯	●	System	(global)	LOGIN_PASSWORD	LOGIN_PASSWORD
▯	●	System	(global)	YOUTUBE_API_KEY	YOUTUBE_API_KEY