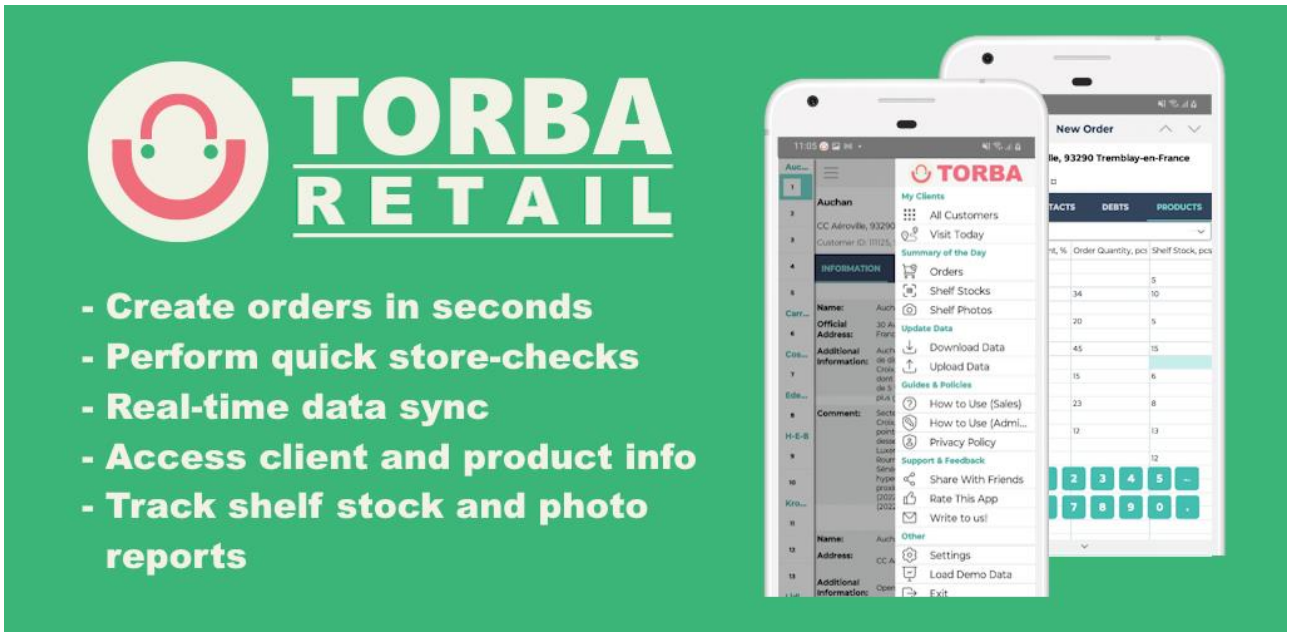
How to set up TORBA Retail



The **<TORBA Retail>** mobile application simplifies the work of a sales representative with their clients. The app enables the sales representative to view the schedule of client visits, the company's product range with photos of items, create orders at points of sale, save GPS coordinates when placing an order, record the company's product stock on store shelves, create photo reports at points of sale, and synchronize data with the company's server in real-time.

To enable the sales representative to work effectively with TORBA Retail, the system administrator must first properly configure the server, prepare all necessary .csv data files, and set the correct parameters in the **<Main menu> => <Settings>** form of the mobile application. Let's go over these simple steps in detail.

1. Prepare your web server for use

The server must be accessible via the internet with the HTTPS protocol enabled, for example*:* **https://mycompany.com**. On the server, create a single working folder that all your sales representatives will use. For example, name this folder **"TORBA"**; then, the working URL will look like: https://mycompany.com/torba.

On the server, create a user with read and write access to the TORBA folder. Configure access permissions for the TORBA folder in the .htaccess file, which should be located in the root of the TORBA folder. For example, it may contain the following content:

```
AuthType Basic

AuthName "Restricted Area"

AuthUserFile "C:/xampp/apache/conf/.htpasswd"

Require valid-user
```

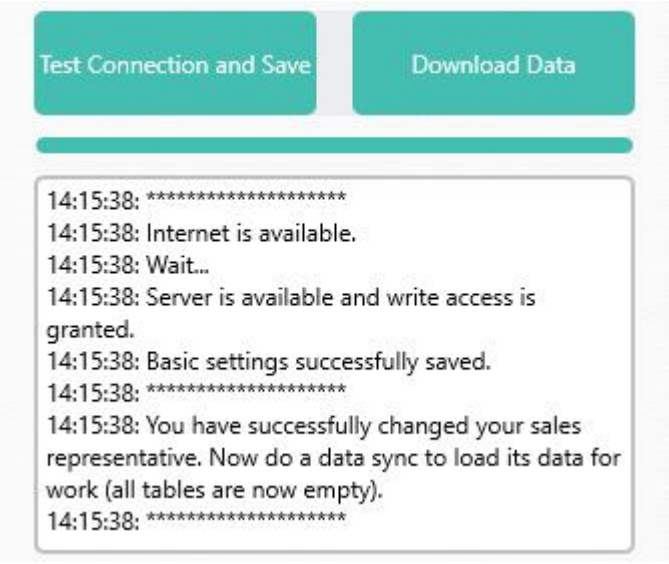
In addition to the .htaccess file, place another file in the root of the TORBA folder, which will allow the sales representative to upload files such as ***orders.csv, orderdetails.csv, stocks.csv, stockdetails.csv, shelfphotos.csv***, and store shelf photos in .jpg format to their working folder. This file is named ***"upload.php"*** and contains the following content:

```
<?php

// Check if the file and representative code exist
```

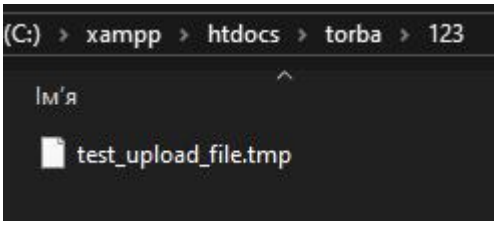
```
if (isset($_FILES['file']) && isset($_POST['rep_code']) &&
isset($_POST['url'])) {
    // Get the representative code from the POST request
    $repCode = $_POST['rep_code'];
    $url = $_POST['url']; // Get the server URL from the POST request
    // Parse the URL to find the base path
    $parsedUrl = parse_url($url);
    // Check if the path is specified
    if (isset($parsedUrl['path'])) {
        $basePath = $parsedUrl['path']; // Get the base path
    } else {
        die("URL does not contain a path."); // If the path is not
specified
    }
    // Form the path to the directory where upload.php is located
    $uploadDir = $_SERVER['DOCUMENT_ROOT'] . $basePath . '/' . $repCode .
'/';
    // Check if the directory exists; if not, create it
    if (!is_dir($uploadDir)) {
        mkdir($uploadDir, 0777, true); // Create the directory with write
permissions
    }
    // Check if the file was uploaded without errors
    if ($_FILES['file']['error'] == UPLOAD_ERR_OK) {
        $tmpFilePath = $_FILES['file']['tmp_name'];
        $newFilePath = $uploadDir . basename($_FILES['file']['name']);
        // Get the file extension
        $fileExtension = strtolower(pathinfo($newFilePath,
PATHINFO_EXTENSION));
        // Allowed extensions
        $allowedExtensions = ['tmp', 'csv', 'jpg', 'jpeg', 'txt', 'pdf',
'png', 'bmp', 'tiff', 'xml', 'json'];
        // Check if the file extension is allowed
        if (in_array($fileExtension, $allowedExtensions)) {
            // Move the file from the temporary directory to the target
directory
            if (move_uploaded_file($tmpFilePath, $newFilePath)) {
                echo "File uploaded successfully.";
            } else {
```

```
                echo "Failed to move the file.";
            }
        } else {
            echo "File type not allowed. Allowed types: " . implode(', ', $allowedExtensions);
        }
    } else {
        echo "Error uploading the file.";
    }
} else {
    echo "File, representative code, or URL was not provided.";
}
?>
```

At this stage, if everything has been set up correctly, you can test the connection to your server from the TORBA Retail mobile application. To do this, launch the application and open the settings form by going to **<Main Menu> => <Settings>**. In the **<Sales Representative ID>** field, enter your representative's internal code in your accounting system (e.g., "123"). In the **<Server Address>** field, enter the URL of your server with the working folder "TORBA" (e.g., **https://mycompany.com/torba**). In the **<User Name>** and **<Password>** fields, enter the test user credentials you created earlier on your server (these credentials are stored in the application in an encrypted form). After that, click the **<Test Connection and Save>** button. If the server is set up correctly, you will see a message confirming the successful connection:

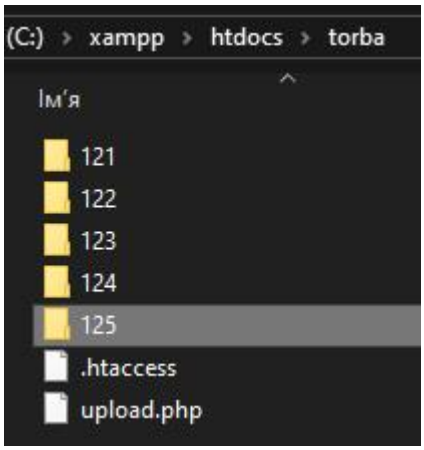


Additionally, a successful connection from the TORBA Retail mobile application to your server is confirmed by the creation of a folder with your sales representative's code (e.g., "123") within the working TORBA folder. In this folder, you will see an empty file named **test_upload_file.tmp**, which verifies that the test user has been correctly granted write permissions. It will look like this:

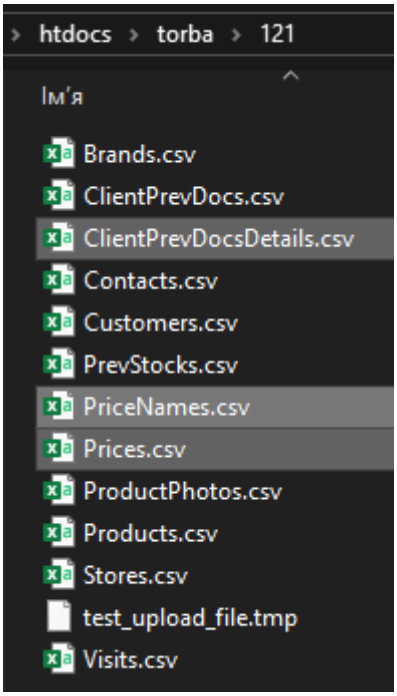


2. Set Up Folders for Your Sales Reps' Data

The next step is to prepare .csv files with the necessary data for each of your sales representatives. Start by creating folders within the TORBA working directory, with names matching the codes of your sales representatives in your accounting system. This way, each sales representative will have their own folder and will work exclusively within it. For example, if you have 5 sales representatives with internal codes **121, 122, 123, 124,** and **125**, then the TORBA folder should contain 5 folders named accordingly:



Here's how to create the CSV data files. All off these files on the server must be encoded in UTF8. The .csv files should use the vertical bar "|" as the field delimiter, so make sure there are no "|" characters in the exported data. Each record must be on a single line - newline (line break) characters within field values are not allowed. A separate set of CSV files is prepared for each sales representative and contains only data relevant to that representative. Each representative's files are saved in a folder named according to their representative code in your accounting system. Thus, each folder contains the following files: **brands.csv, products.csv, customers.csv, contacts.csv, stores.csv, visits.csv, clientprevdocs.csv, prevstocks.csv**, and **productphotos.csv**. Additionally, if the subscription is active, the following files should also be included: **prices.csv, pricenames.csv,** and **clientprevdocsdetails.csv.**



Each CSV file follows a specific structure, where each column has a defined name and data type. In the exported data, all numeric values should be without thousand separators, and the decimal separator must be a period, e.g., 12345.12. For date fields, use the format dd-mm-yyyy (or dd-mm-yyyy hh:mm). Based on this information, prepare the following CSV files:

==========

1. Brands.csv (contains a list of the brands you sell).

Example:

BrandID|BrandName

1111|Toblerone

1112|Ferrero Rocher

Field types:

BrandID => Integer NOT NULL;

BrandName => Text NOT NULL.

==========

2. Contacts.csv (contains a list of client's store contacts you work with).

Example:

ContactID|StoreID|Name|Role|Phone|Email|AdditionalInfo|Comment

1|222254|Ellen Ripley|Manager|(555) 012-3451|manager001@mycompany.com|Available for orders Mon-Fri 9:00-17:00; born: 1985-07-22.|Friendly, but prefers detailed order descriptions.

2|222256|Leia Organa|Manager|(555) 012-3452|manager002@mycompany.com|Handles frozen foods and beverages; speaks English and French; born: 1979-12-05.|Requires prior confirmation for large orders.

Field types:

ContactID, StoreID => Integer NOT NULL;

Name, Role, Phone, Email, AdditionalInfo, Comment => Text.

Relationships with other tables:

FOREIGN KEY (StoreID) REFERENCES Stores(StoreID).

==========

3. Customers.csv (contains a list of your customers).

Example:

CustomerID|CustomerName|Address|AdditionalInfo|Comment

111111|Walmart Supercenter|702 SW 8th St, Bentonville, AR 72716, USA|Additional information about Walmart|Comment about Walmart

111112|Kroger|1014 Vine St, Cincinnati, OH 45202, USA|Additional information about Kroger|Comment about Kroger

Field types:

CustomerID => Integer NOT NULL;

CustomerName => Text NOT NULL;

Address, AdditionalInfo, Comment => Text.

Note: one customer can have one or more stores with different addresses (see stores.csv).

==========

4. Stores.csv (contains a list of your customers' stores).

Example:

StoreID|StoreName|CustomerID|StoreAddress|GPSCoordinates|AdditionalInfo|Comment

222221|Walmart Boll Weevil Cir|111111|600 Boll Weevil Cir, Enterprise, AL 36330|31.315717793368318,-85.82743997454594|Additional information about Walmart Boll Weevil Cir|Comment about Walmart Boll Weevil Cir

222222|Walmart Nc Highway|111111|300 Nc Highway 24, Morehead City, NC 28557|34.7352190678906,-76.8108006658345|Additional information about Walmart Nc Highway|Comment about Walmart Nc Highway

Field types:

StoreID, CustomerID => Integer NOT NULL;

StoreName => Text NOT NULL;

StoreAddress, GPSCoordinates, AdditionalInfo, Comment => Text.

Relationships with other tables:

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID).

==========

5. ClientPrevDocs.csv (contains the client's orders for the previous period: unpaid invoices, previous orders).

Example:

DocID|CustomerID|StoreID|RepID|DocumentNumber|InvoiceNumber|SaleDate|SaleAmount|PaidAmount|LastPaymentDate|PaymentDelay|OverdueDays

5125706|111111|222221|112233|5125706|23098632|04-02-2025|3482.37|49.52|03-04-2025|40|24

5125606|111111|222222|112233|5125606|23091880|12-02-2025|4835.81|107.49|14-02-2025|37|19

Field types:

DocID, CustomerID, StoreID, RepID => Integer NOT NULL;

PaymentDelay, OverdueDays => Integer;

DocumentNumber, InvoiceNumber, SaleDate, LastPaymentDate => Text;

SaleAmount, PaidAmount => Real.

Relationships with other tables:

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID);

FOREIGN KEY (StoreID) REFERENCES Stores(StoreID).

Note: PaymentDelay is the payment delay in days, OverdueDays indicates how many days the payment is overdue, and RepID is the sales representative’s code. If multiple sales representatives visit this store, it is better to export unpaid sales documents for all these

representatives. The data from this file and the next one will be used by the sales representative to view the customer's accounts receivable and to review their previous order when creating a new one.

==========

6. ClientPrevDocsDetails.csv (contains detailed product data from previous customer orders).

Example:

DocID|ProductID|Quantity|DiscountPercent

5125706|1234556|34|2.5

5125706|1234556|78|3

Field types:

DocID, ProductID => Integer NOT NULL;

Quantity, DiscountPercent => Real.

Relationships with other tables:

FOREIGN KEY (DocID) REFERENCES ClientPrevDocs(DocID);

FOREIGN KEY (ProductID) REFERENCES Products(ProductID).

Note: ProductID – product item code, Quantity – ordered quantity of the product, DiscountPercent – discount applied to the product.

7. Products.csv (contains a list of your product items grouped by the brands you sell).

Example:

ProductID|ProductName|BrandID|QuantityInStock|Price|MaxDiscountPercent|Barcode|Description|UnitsPerCase|UnitsPerPallet|Weight|Volume

1234531|Toblerone Golden Caramel 360g|1111|11892|17.99|0|2515348638719|Toblerone Golden Caramel 360g Description|80|1320|360|0

1234537|Toblerone Dark Bar 360G|1111|41200|3.6|0|8918163343396|Toblerone Dark Bar 360G Description|20|1320|360|0

Field types:

ProductID => Integer NOT NULL;

BrandID => Integer;

ProductName => Text NOT NULL;

Barcode, Description => Text;

QuantityInStock, Price, MaxDiscountPercent, UnitsPerCase, UnitsPerPallet, Weight, Volume => Real.

Relationships with other tables:

FOREIGN KEY (BrandID) REFERENCES Brands(BrandID).

Note: QuantityInStock – the available quantity of the product in your warehouse. Price – the base price of the product (used in the free version; also used as a fallback in the subscription version if the Prices.csv file is empty or does not contain information about the

product). MaxDiscountPercent is the maximum allowable discount on an item.

==========

8. Visits.csv (contains the customer visit schedule for the sales representative).

Example:

StoreID|RepID|VisitDate

222221|121|2024-11-05

222221|121|2024-11-08

222223|121|2024-11-05

Field types:

StoreID, RepID => Integer NOT NULL;

VisitDate => Text.

Relationships with other tables:

FOREIGN KEY (StoreID) REFERENCES Stores(StoreID).

Note: RepID is the sales representative’s code. This file exports the schedule of client visits for the sales representative (exporting this data at least one week ahead, so the representative knows which clients to visit tomorrow, the day after, and so on).

==========

9. PrevStocks.csv (contains data about the stock levels on the customer’s store shelves during the previous visit).

Example:

StockID|StoreID|RepID|ProductID|Quantity|StockDate

104|222223|112233|1234550|30|2024-10-20

105|222223|112233|1234551|24|2024-10-20

Field types:

StockID, StoreID, RepID, ProductID => Integer NOT NULL;

Quantity => Real;

StockDate => Text.

Relationships with other tables:

FOREIGN KEY (StoreID) REFERENCES Stores(StoreID);

FOREIGN KEY (ProductID) REFERENCES Products(ProductID).

Note: The file exports data on previous product stock levels on store shelves along the sales representative's route. This historical data is used by the sales representative when creating a new order, as well as for generating a calculated (suggested) order. StockID and StockDate refer to the code and date of the document containing information about product stock levels on store shelves. RepID is the sales representative’s code.

==========

10. ProductPhotos.csv (contains photos of your product items).

Example:

ProductID|FullPhoto

1234531|iVBORw0KGgoAAAANSUh...

1234537|iVBORw0KGgoAAAANSUh...

Field types:

ProductID => Integer NOT NULL;

FullPhoto => Blob encoded in Base64.

Relationships with other tables:

FOREIGN KEY (ProductID) REFERENCES Products(ProductID).

Note: In the free version, only one photo of each product item is displayed. Due to the large data volume, avoid storing unnecessary data in this table to speed up synchronization.

==========

11. PriceNames.csv (contains a list of your price lists that the sales representative can use when creating a new order).

Example:

PriceID|PriceName|Currency

101|Base Price|USD

102|Wholesale Price|USD

103|Service Station Price|USD

104|Premium Retail Price|EUR

Field types:

PriceID => Integer NOT NULL;

PriceName, Currency => Text NOT NULL.

==========

12. Prices.csv (contains the prices of your products by price lists).

Example:

ProductID|PriceID|Price

1234531|101|17.99

1234531|102|16.19

1234531|103|19.79

1234531|104|21.59

Field types:

ProductID, PriceID => Integer NOT NULL;

Price => Real NOT NULL.

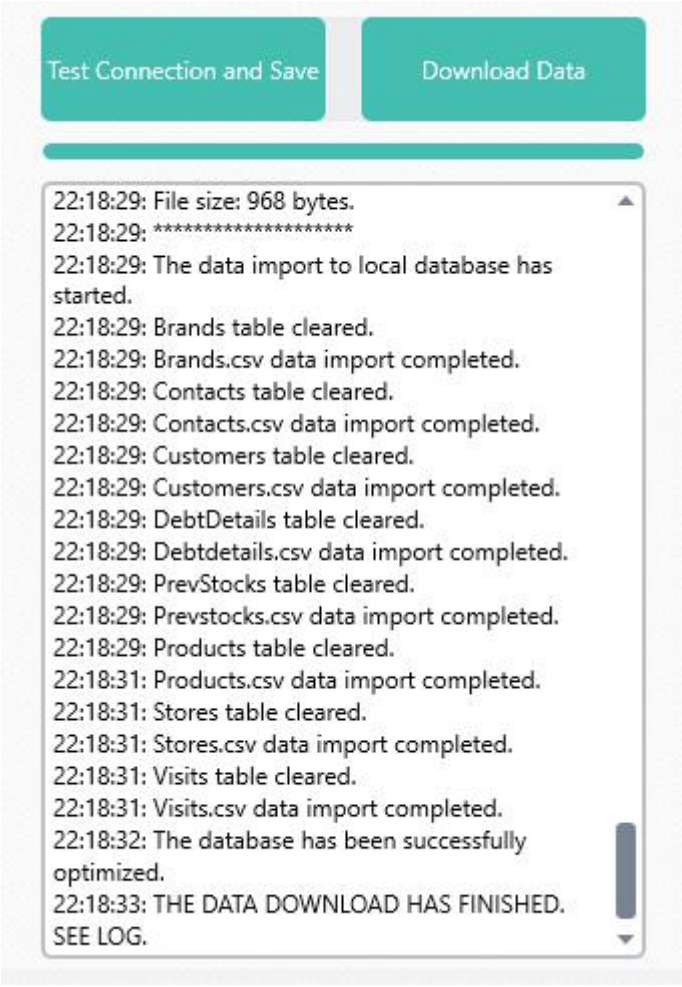
Relationships with other tables:

```
FOREIGN KEY (ProductID) REFERENCES Products(ProductID);

FOREIGN KEY (PriceID) REFERENCES PriceNames(PriceID);

Note: the file contains the product prices based on different price lists
(PriceID) for each product (ProductID).

======================
```

Of course, it is not necessary to do all this manually; it is better to create a script that runs, for example, every night, selects data from your accounting system, generates all the necessary .csv files for all sales representatives, and saves them in the appropriate folders on your server.

3. Testing Data Synchronization in the TORBA Retail App

After setting up the working server and generating the CSV data files, try updating the data in the TORBA Retail app for one of your sales representatives. To do this, go to the settings screen via **<Main Menu> => <Settings>**, verify that the information in the fields **<Sales Representative ID>**, **<Server Address>**, **<User Name>**, and **<Password>** is correct, then click **<Test Connection and Save>** again, followed by **<Download Data>**. The data download from your server will start immediately, with each synchronization step displayed in the log at the bottom of the screen.

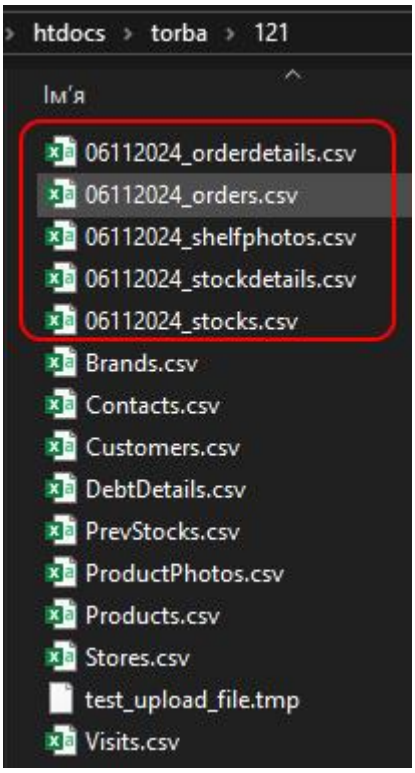


If the data download is successful, you will need to repeat this procedure on each sales representative's mobile device. Remember to enter the appropriate sales representative code in the **<Sales Representative ID>** field on the **<Settings>** screen. Each sales representative has a unique code.

4. Understanding the Structure of Sales Rep CSV Files

Now let's look at the structure of the CSV files that your sales representatives will create during their daily work. Your sales reps visit clients and generate new documents, such as orders (orders.csv and

orderdetails.csv), stock levels of products on store shelves (**stocks.csv** and **stockdetails.csv**), and photos of store shelves with your products (**shelfphotos.csv**). They upload all of these documents to your company's server as CSV files. These files are saved in each sales representative's working folder, named according to their code in your accounting system (this code is also set in the application under **<Main Menu> => <Settings>**). Each file name begins with the current date in the format ddmmyyyy_, for example, 06112024_orders.csv. This looks as follows:



The CSV files uploaded by the sales rep are overwritten with each data upload and contain all documents created since the beginning of the workday. The sales representative cannot edit documents in the TORBA Retail app that have already been sent to the server. This way, you can import this data into your accounting system for further processing.

Let's look at the structure of the files: **orders.csv, orderdetails.csv, stocks.csv, stockdetails.csv**, and **shelfphotos.csv**.

==========

1. Orders.csv (contains order document headers).

Example:

```
OrderID|StoreID|RepID|PriceID|OrderDate|TotalAmount|Comment|GPSCoord

1|222253|121|104|06-11-2024 12:04:12|484.65|Deliver the goods tomorrow after 3:00 PM.|31.315717793368318,-85.82743997454594
```

Field types:

OrderID, StoreID, RepID, PriceID => Integer;

OrderDate, Comment, GPSCoord => Text;

TotalAmount => Real.

Relationships with other tables:

FOREIGN KEY (StoreID) REFERENCES Stores(StoreID);

FOREIGN KEY (PriceID) REFERENCES PriceNames(PriceID).

Note: Here, RepID is the sales representative’s code. GPSCoord - the coordinates of the location where the order was created. PriceID - the identifier of the price list on which the order is based. It is available only with a subscription; in the free version, PriceID remains empty.

==========

2. OrderDetails.csv (contains order document details).

Example:

OrderID|ProductID|Quantity|DiscountPercent

1|1234537|20|1.5

1|1234551|15|1.5

Field types:

OrderID, ProductID => Integer;

Quantity, DiscountPercent => Real.

Relationships with other tables:

FOREIGN KEY (ProductID) REFERENCES Products(ProductID);

FOREIGN KEY (OrderID) REFERENCES Orders(OrderID).

==========

3. Stocks.csv (contains stock document headers).

Example:

StockID|StoreID|RepID|StockDate|GPSCoord

1|222253|121|06-11-2024 12:04:12|31.315717793368318,-85.82743997454594

Field types:

StockID, StoreID, RepID => Integer;

StockDate, GPSCoord => Text.

Relationships with other tables:

FOREIGN KEY (StoreID) REFERENCES Stores(StoreID).

Note: the file includes data on stock levels on the customer’s store shelves during the current visit. RepID is the sales representative's code.

==========

4. StockDetails.csv (contains stock document details).

Example:

StockID|ProductID|Quantity

1|1234537|3

1|1234551|4

Field types:

StockID, ProductID => Integer;

Quantity => Real.

Relationships with other tables:

FOREIGN KEY (StockID) REFERENCES Stocks(StockID);

FOREIGN KEY (ProductID) REFERENCES Products(ProductID).

==========

5. Shelfphotos.csv (contains photos of your product on the store shelves).

Example:

StoreID|RepID|PhotoDate|PhotoFull

222253|121|06-11-2024 12:04:12|iVBORw0KGgoAAAANSUh...

Field types:

StoreID, RepID => Integer;

PhotoDate => Text;

PhotoFull => Blob encoded in Base64.

Relationships with other tables:

FOREIGN KEY (StoreID) REFERENCES Stores(StoreID).

Note: RepID is the sales representative's code.

==========

Note: When sales representatives upload store shelf photos to the server, the **shelfphotos.csv** file is accompanied by the actual shelf photos saved as jpg files in the representative's working folder. Each photo file name includes the current date and the store code (for example, **06112024_12345.jpg**).

5. Centralizing Daily Order & Stock Reports

At this stage, all order documents and shelf stock records from your field sales representatives are stored in separate subfolders—one per representative—within the main **Torba** folder on your server.

If you wish to **combine these files** (separately for orders and stock) into unified CSV files for easier import into your accounting or ERP system, follow these steps:

1. **Create a PHP script** on your server named **combine_csv.php** with the following content (see script below).
2. **Schedule this script** to run automatically several times per day using a task scheduler (e.g., Task Scheduler on Windows or cron on Linux).

As a result, the script will generate the following **merged output files** in the root of the **Torba** folder:

- ddmmyyyy_all_orders.csv
- ddmmyyyy_all_orderdetails.csv
- ddmmyyyy_all_stocks.csv
- ddmmyyyy_all_stockdetails.csv

Each of these files will contain **combined data from all your sales reps**, making the integration process into your system faster and more efficient.

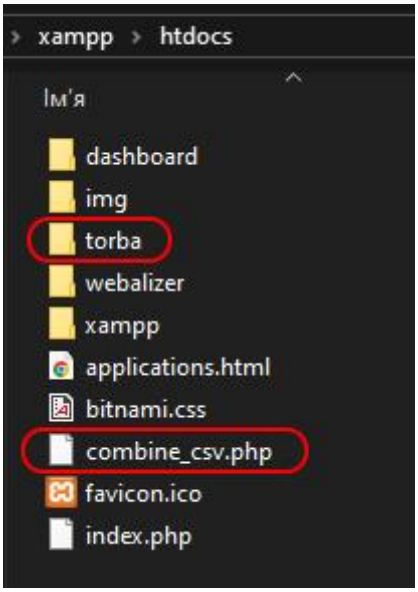
```
<?php
// Get the current date in ddmmyyyy format
```

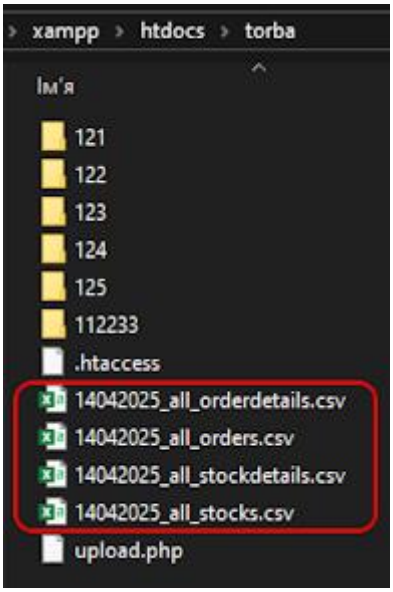
```
$date = date('dmY');
// Base directory
$baseDir = __DIR__ . '/torba';

// Output file paths
$allOrdersFile = $baseDir . "/{$date} all orders.csv";
$allOrderDetailsFile = $baseDir . "/{$date} all orderdetails.csv";
$allStocksFile = $baseDir . "/{$date} all stocks.csv";
$allStockDetailsFile = $baseDir . "/{$date} all stockdetails.csv";
// Open output files for writing
$ordersOutput = fopen($allOrdersFile, 'w');
$orderDetailsOutput = fopen($allOrderDetailsFile, 'w');
$stocksOutput = fopen($allStocksFile, 'w');
$stockDetailsOutput = fopen($allStockDetailsFile, 'w');
// Flags to ensure headers are written only once
$ordersHeaderWritten = false;
$orderDetailsHeaderWritten = false;
$stocksHeaderWritten = false;
$stockDetailsHeaderWritten = false;
// Get list of subdirectories in torba
$dirs = array_filter(glob($baseDir . '/*'), 'is_dir');
// Loop through each subdirectory
foreach ($dirs as $dir) {
    // File paths
    $ordersFile = $dir . "/{$date}_orders.csv";
    $orderDetailsFile = $dir . "/{$date}_orderdetails.csv";
    $stocksFile = $dir . "/{$date}_stocks.csv";
    $stockDetailsFile = $dir . "/{$date}_stockdetails.csv";
    // Process orders
    if (file_exists($ordersFile)) {
        $handle = fopen($ordersFile, 'r');
        if ($handle) {
            $lineNum = 0;
            while (($line = fgetcsv($handle, 0, '|')) !== false) {
                if ($lineNum === 0 && !$ordersHeaderWritten) {
                    fputcsv($ordersOutput, $line, '|');
```

```
                    $ordersHeaderWritten = true;
                } elseif ($lineNum > 0) {
                    fputcsv($ordersOutput, $line, '|');
                }
                $lineNum++;
            }
            fclose($handle);
        }
    }
    // Process orderdetails
    if (file_exists($orderDetailsFile)) {
        $handle = fopen($orderDetailsFile, 'r');
        if ($handle) {
            $lineNum = 0;
            while (($line = fgetcsv($handle, 0, '|')) !== false) {
                if ($lineNum === 0 && !$orderDetailsHeaderWritten) {
                    fputcsv($orderDetailsOutput, $line, '|');
                    $orderDetailsHeaderWritten = true;
                } elseif ($lineNum > 0) {
                    fputcsv($orderDetailsOutput, $line, '|');
                }
                $lineNum++;
            }
            fclose($handle);
        }
    }
    // Process stocks
    if (file_exists($stocksFile)) {
        $handle = fopen($stocksFile, 'r');
        if ($handle) {
            $lineNum = 0;
            while (($line = fgetcsv($handle, 0, '|')) !== false) {
                if ($lineNum === 0 && !$stocksHeaderWritten) {
                    fputcsv($stocksOutput, $line, '|');
                    $stocksHeaderWritten = true;
                } elseif ($lineNum > 0) {
```

```
                    fputcsv($stocksOutput, $line, '|');
                }
                $lineNum++;
            }
            fclose($handle);
        }
    }
    // Process stockdetails
    if (file_exists($stockDetailsFile)) {
        $handle = fopen($stockDetailsFile, 'r');
        if ($handle) {
            $lineNum = 0;
            while (($line = fgetcsv($handle, 0, '|')) !== false) {
                if ($lineNum === 0 && !$stockDetailsHeaderWritten) {
                    fputcsv($stockDetailsOutput, $line, '|');
                    $stockDetailsHeaderWritten = true;
                } elseif ($lineNum > 0) {
                    fputcsv($stockDetailsOutput, $line, '|');
                }
                $lineNum++;
            }
            fclose($handle);
        }
    }
}
// Close all output files
fclose($ordersOutput);
fclose($orderDetailsOutput);
fclose($stocksOutput);
fclose($stockDetailsOutput);

echo "Merging completed successfully!\n";
?>
```

Here's how it looks on the server:





Remember: each sales representative downloads new data from their own directory on the server, which is named after their representative code in your accounting system. All documents created by the sales representative are uploaded to their designated directory on the server. Your task is to write a script that automates the process of updating the CSV files for all your sales representatives and imports the data they upload into your accounting system.

For any questions or suggestions regarding the TORBA Retail mobile application, please contact us at **torba.retail@gmail.com**.