ДОДАТОК А

Код програми для обробки CRUD та оновлення замовлень

CRUD в адмін панелі

```
<?php

namespace app\controllers\admin;

use app\models\Product;
use Yii;
use yii\db\StaleObjectException;
use yii\filters\auth\CompositeAuth;
use yii\filters\auth\HttpBearerAuth;
use yii\rest\Controller;
//popular items
use app\models\PopularForm;
use app\features\popular\usecase\GetAllPopularUseCase;
use app\features\popular\usecase\UpdatePopularUseCase;
use app\features\popular\usecase\CreatePopularUseCase;
use app\features\popular\usecase\DeletePopularUseCase;
use app\features\popular\usecase\SetByStorageImageUseCase;
use app\features\popular\usecase\UploadImageIdPopularUseCase;
//product items
use app\models\ProductForm;
use app\features\product\usecase\GetAllProductsUseCase;
use app\features\product\usecase\UploadImageIdProductUseCase;
use app\features\product\usecase\DeleteProductUseCase;
use app\features\product\usecase\CreateProductUseCase;
use app\features\product\usecase\UpdateProductUseCase;
use app\features\product\usecase\SetByStorageImageProductUseCase;
use app\features\product\usecase\UpdateProductVisibleUseCase;
//stocks items
use app\models\StockForm;
use app\features\stocks\usecase\GetAllStocksUseCase;
use app\features\stocks\usecase\DeleteStocksUseCase;
use app\features\stocks\usecase\CreateStocksUseCase;
use app\features\stocks\usecase\UpdateStocksUseCase;
use app\features\stocks\usecase\UploadImageIdStocksUseCase;
use app\features\stocks\usecase\SetByStorageImageStockUseCase;
use app\features\product\usecase\UpdateProductPriceWithDiscount;

class AdminController extends Controller
{
//  popular items
    private GetAllPopularUseCase $allPopularUseCase;
    private UpdatePopularUseCase $updatePopularUseCase;
    private CreatePopularUseCase $createPopularUseCase;
    private SetByStorageImageUseCase $setByStorageImageUseCase;
```

```
    private UploadImageIdPopularUseCase $uploadImageIdPopularUseCase;
    private DeletePopularUseCase $deletePopularUseCase;
//  product items
    private GetAllProductsUseCase $allProductsUseCase;
    private DeleteProductUseCase $deleteProductUseCase;
    private CreateProductUseCase $createProductUseCase;
    private UpdateProductUseCase $updateProductUseCase;
    private UploadImageIdProductUseCase $uploadImageIdProductUseCase;
    private SetByStorageImageProductUseCase
$setByStorageImageProductUseCase;
    private UpdateProductVisibleUseCase $updateProductVisibleUseCase;
//  stocks items
    private GetAllStocksUseCase $allStocksUseCase;
    private DeleteStocksUseCase $deleteStocksUseCase;
    private CreateStocksUseCase $createStocksUseCase;
    private UpdateStocksUseCase $updateStocksUseCase;
    private UploadImageIdStocksUseCase $uploadImageIdStocksUseCase;
    private SetByStorageImageStockUseCase
$setByStorageImageStockUseCase;
    private UpdateProductPriceWithDiscount
$updateProductPriceWithDiscount;



    public function __construct($id, $module,
    //  popular items
        GetAllPopularUseCase $allPopularUseCase,
        CreatePopularUseCase $createPopularUseCase,
        UpdatePopularUseCase $updatePopularUseCase,
        SetByStorageImageUseCase $setByStorageImageUseCase,
        UploadImageIdPopularUseCase $uploadImageIdPopularUseCase,
        DeletePopularUseCase $deletePopularUseCase,
    //  product items
        GetAllProductsUseCase $allProductsUseCase,
        DeleteProductUseCase $deleteProductUseCase,
        CreateProductUseCase $createProductUseCase,
        UpdateProductUseCase $updateProductUseCase,
        UploadImageIdProductUseCase $uploadImageIdProductUseCase,
        SetByStorageImageProductUseCase
$setByStorageImageProductUseCase,
        UpdateProductVisibleUseCase $updateProductVisibleUseCase,
    //  stocks items
        GetAllStocksUseCase $allStocksUseCase,
        DeleteStocksUseCase $deleteStocksUseCase,
        CreateStocksUseCase $createStocksUseCase,
        UpdateStocksUseCase $updateStocksUseCase,
        UploadImageIdStocksUseCase $uploadImageIdStocksUseCase,
        SetByStorageImageStockUseCase $setByStorageImageStockUseCase,
        UpdateProductPriceWithDiscount $updateProductPriceWithDiscount

    )
```

```
    {
        parent::__construct($id, $module);
    //  popular items
        $this->allPopularUseCase = $allPopularUseCase;
        $this->createPopularUseCase = $createPopularUseCase;
        $this->updatePopularUseCase = $updatePopularUseCase;
        $this->setByStorageImageUseCase = $setByStorageImageUseCase;
        $this->uploadImageIdPopularUseCase=
$uploadImageIdPopularUseCase;
        $this->deletePopularUseCase= $deletePopularUseCase;
    //  product items
        $this->allProductsUseCase = $allProductsUseCase;
        $this->deleteProductUseCase=$deleteProductUseCase;
        $this->createProductUseCase=$createProductUseCase;
        $this->updateProductUseCase=$updateProductUseCase;
        $this-
>uploadImageIdProductUseCase=$uploadImageIdProductUseCase;
        $this-
>setByStorageImageProductUseCase=$setByStorageImageProductUseCase;
        $this-
>updateProductPriceWithDiscount=$updateProductPriceWithDiscount;
        $this-
>updateProductVisibleUseCase=$updateProductVisibleUseCase;
    //  stocks items
        $this->allStocksUseCase = $allStocksUseCase;
        $this->deleteStocksUseCase=$deleteStocksUseCase;
        $this->createStocksUseCase=$createStocksUseCase;
        $this->updateStocksUseCase=$updateStocksUseCase;
        $this->uploadImageIdStocksUseCase=$uploadImageIdStocksUseCase;
        $this-
>setByStorageImageStockUseCase=$setByStorageImageStockUseCase;

    }

    public function behaviors(): array
    {
        $behaviors = parent::behaviors();
        $behaviors['authenticator'] = [
            'class' => CompositeAuth::class,
            'authMethods' => [
                HttpBearerAuth::class,
            ],
        ];
        return $behaviors;
    }

    public function actionIndex():array {
        return [

            'products' => $this->allProductsUseCase->execute(),
```

```
//            'populars' => $this->allPopularUseCase->execute(),
        ];
    }
    public function actionGetStocks():array {
        return [
            'stocks' => $this->allStocksUseCase->execute(),
        ];
    }

    public function actionDeletePopular(): array
    {
        $popularId = Yii::$app->request->post('id');

        if (empty($popularId)) {
            return [
                'error' => true,
                'delete' => false,
            ];
        }

        $status = $this->deletePopularUseCase->execute($popularId);

        if ($status) {
            return [
                'error' => false,
                'delete' => true,
            ];
        }

        return [
            'error' => true,
            'delete' => false,
        ];
    }

    public function actionInsertPopular(): array
    {
        $model = new PopularForm();

        if ($model->load(Yii::$app->request->post(), '') && $model-
>validate() || Yii::$app->request->isPost) {
            extract(Yii::$app->request->post());
            $image = $this->setByStorageImageUseCase->execute();
            $this->createPopularUseCase->execute($title, $description,
$price,$image);

            return [
                'error' => false,
                'send' => true,
            ];
        }
```

```
            return [
                'error' => true,
                'send' => false,
            ];
    }

    public function actionUpdatePopular(): array
    {
        $model = new PopularForm();
        $status = false;
        if ($model->load(Yii::$app->request->post(), '') && $model-
>validate() || Yii::$app->request->isPost) {
            extract(Yii::$app->request->post());
            $this->updatePopularUseCase->execute($id, $title,
$description, $price);
            $path = $this->setByStorageImageUseCase->execute();
            $id = Yii::$app->request->post('id');
            $id = $id ? trim($id) : '';
            $status = $this->uploadImageIdPopularUseCase-
>execute($path, $id);
        }
            if ($status) {
                return [
                    'error' => false,
                    'send' => true,
                ];
            }
            return [
                'error' => true,
                'send' => false,
            ];

    }
    /**
     * @throws StaleObjectException
     * @throws \Throwable
     */
    public function actionDeleteProduct(): array
    {
        $productId = Yii::$app->request->post('product_id');

        if (empty($productId)) {
            return [
                'error' => true,
                'delete' => false,
            ];
        }

        $status = $this->deleteProductUseCase->execute($productId);

        if ($status) {
```

```
            return [
                'error' => false,
                'delete' => true,
            ];
        }

        return [
            'error' => true,
            'delete' => false,
        ];
    }

    public function actionInsertProduct(): array
    {
        $model = new ProductForm();

        if ($model->load(Yii::$app->request->post(), '') && $model-
>validate() || Yii::$app->request->isPost) {
            extract(Yii::$app->request->post());
            $image = $this->setByStorageImageProductUseCase-
>execute();
            $status = $this->createProductUseCase->execute($title,
$description, $price, $image, $category_id);
            if ($status && $image) {
                return [
                    'error' => false,
                    'send' => true,
                ];
            }
        }
            return [
                'error' => true,
                'send' => false,
            ];
    }

    public function actionUpdateProduct(): array
    {
        $model = new Product();
        $status = false;
        if ($model->load(Yii::$app->request->post(), '') && $model-
>validate() || Yii::$app->request->isPost) {
            extract(Yii::$app->request->post());

            if (!empty($image)) {
                $status = true;
                $this->updateProductUseCase->execute($product_id,
$title, $description, $price, $image);
            } else {
                $path = $this->setByStorageImageProductUseCase-
>execute();
```

```
                if ($path === false) {
                    return [
                        'error' => true,
                        'send' => false,
                        'status' => false,
                        'message' => 'Invalid image file.'
                    ];
                } else {
                    $id = Yii::$app->request->post('product_id');
                    $id = $id ? trim($id) : '';
                    $status = $this->uploadImageIdProductUseCase-
>execute($path, $id);
                }
            }
        }
        if ($status) {
            return [
                'error' => false,
                'send' => true,
                'status' => $status,
            ];
        }
        return [
            'error' => true,
            'send' => false,
            'status' => $status,
        ];
    }


    public function actionDeleteStock(): array
    {
        $stock_id = Yii::$app->request->post('stock_id');

        if (empty($stock_id)) {
            return [
                'error' => true,
                'delete' => false,
                "id"=> $stock_id
            ];
        }

        $status = $this->deleteStocksUseCase->execute($stock_id);


        if ($status) {
            return [
                'error' => false,
                'delete' => true,
                'status' => $status
```

```
            ];
        }

        return [
            'error' => true,
            'delete' => false,
        ];
    }
    public function actionInsertStock(): array
    {
        $model = new StockForm();

        if ($model->load(Yii::$app->request->post(), '') && $model-
>validate() || Yii::$app->request->isPost) {
            extract(Yii::$app->request->post());
            $image = $this->setByStorageImageStockUseCase->execute();
            $status = $this->createStocksUseCase->execute($image,
$product_id);
            if ($status && $image) {
                return [
                    'error' => false,
                    'send' => true,
                    'image' => $image,
                    'status'=> $status
                ];
            }
        }
        return [
            'error' => true,
            'send' => false,
            'image' => $image,
            'status'=> $status
        ];
    }

    public function actionUpdateVisible(): array
    {
        $model = new StockForm();

        if ($model->load(Yii::$app->request->post(), '') && $model-
>validate() || Yii::$app->request->isPost) {
            extract(Yii::$app->request->post());
            $this->updateProductVisibleUseCase->execute($product_id,
$visible);
            return [
                'error' => false,
                'send' => true,
            ];
        }
        return [
            'error' => true,
```

```
            'send' => false,
        ];
    }



    public function actionUpdateStock(): array
    {
        $model = new StockForm();
        $status = false;
        if ($model->load(Yii::$app->request->post(), '') && $model-
>validate() || Yii::$app->request->isPost) {
            extract(Yii::$app->request->post());

            if (!empty($image)) {
                $status = true;
                $this->updateStocksUseCase->execute($stock_id,
$discount, $image);
                $this->updateProductPriceWithDiscount-
>execute($product_id, $discount);
            } else {
                $path = $this->setByStorageImageStockUseCase-
>execute();
                $this->updateProductPriceWithDiscount-
>execute($product_id, $discount);

                if ($path === false) {
                    return [
                        'error' => true,
                        'send' => false,
                        'status' => false,
                        'message' => 'Invalid image file.'
                    ];
                } else {
                    $id = Yii::$app->request->post('stock_id');
                    $id = $id ? trim($id) : '';
                    $status = $this->uploadImageIdStocksUseCase-
>execute($path, $id);
                    $this->updateProductPriceWithDiscount-
>execute($product_id, $discount);
                }
            }
        }
        if ($status) {
            return [
                'error' => false,
                'send' => true,
                'status' => $status,
            ];
        }
```

```
        return [
            'error' => true,
            'send' => false,
            'status' => $status,
        ];
    }



}
```

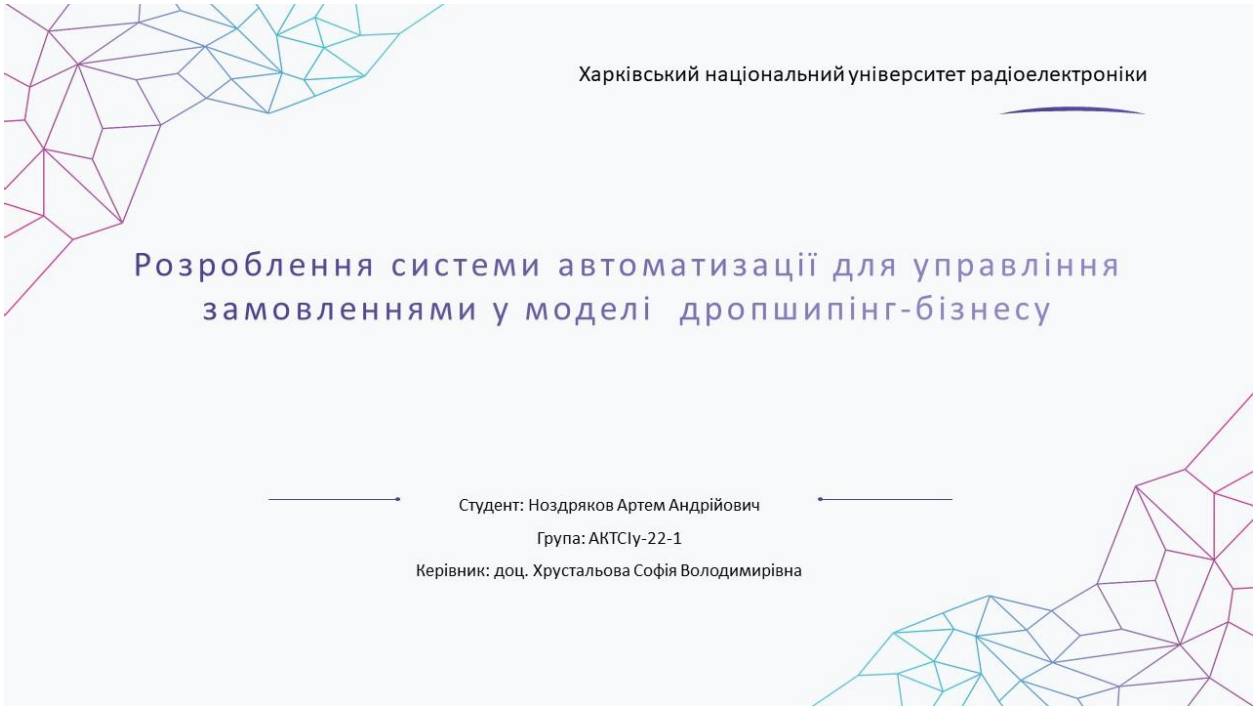
Відправка замовлення постачальнику через API

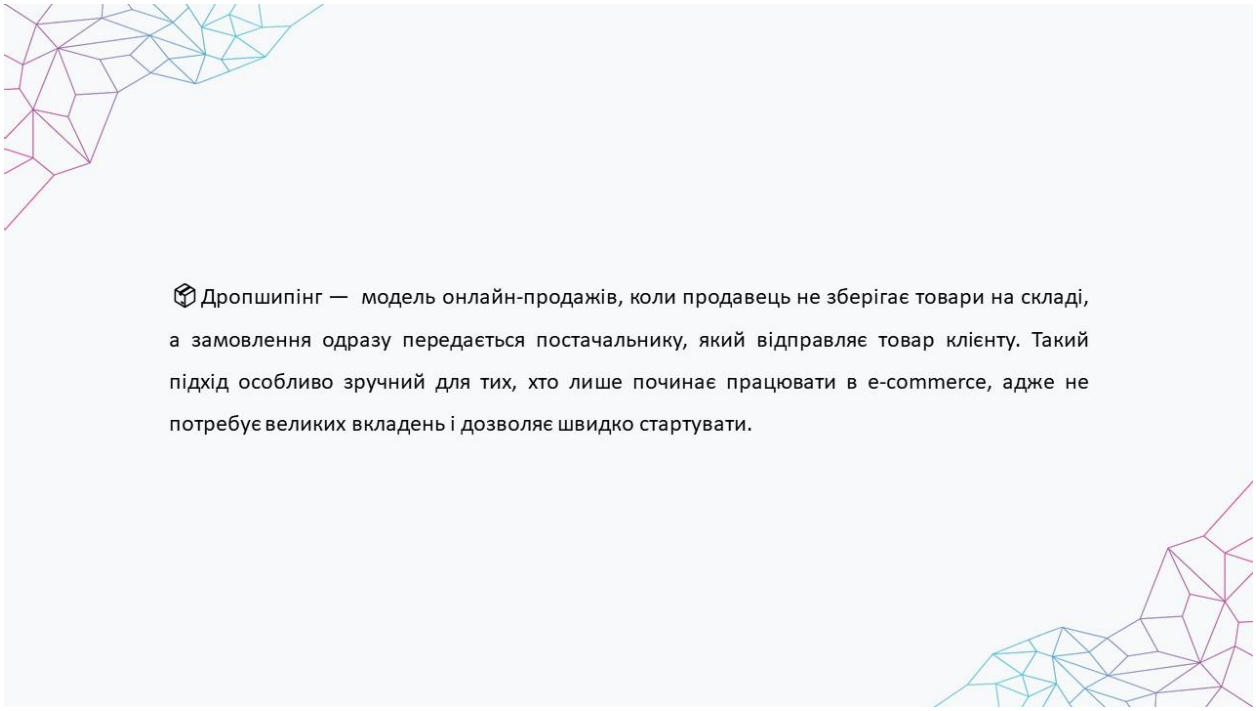
```
namespace app\components;

use Yii;
use yii\httpclient\Client;
use app\models\Order;

class SupplierApiService
{
    public static function sendOrderToSupplier(Order $order): bool
    {
        $client = new Client();

        $product = $order->product;
        $supplier = $product->supplier;

        $response = $client->createRequest()
            ->setMethod('POST')
            ->setUrl($supplier->api_endpoint . '/orders')
            ->setHeaders(['Authorization' => 'Bearer ' . $supplier-
>api_token])
            ->setData([
                'order_id' => $order->id,
                'product' => [
                    'name' => $product->name,
                    'sku' => $product->id,
                    'quantity' => 1,
                ],
                'customer' => [
                    'name' => $order->customer_name,
                    'email' => $order->customer_email,
                    'phone' => $order->customer_phone,
                    'address' => $order->delivery_address,
                ],
                'delivery_method' => $order->delivery_type,
            ])
            ->send();

        if ($response->isOk) {
```

```
            Yii::info("Замовлення #{$order->id} успішно передано
постачальнику", 'supplier');
            return true;
        } else {
            Yii::error("Помилка при надсиланні замовлення #{$order-
>id}: " . $response->content, 'supplier');
            return false;
        }
    }
}
```

ДОДАТОК Б

Демонстраційний матеріал

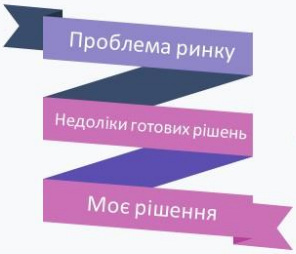




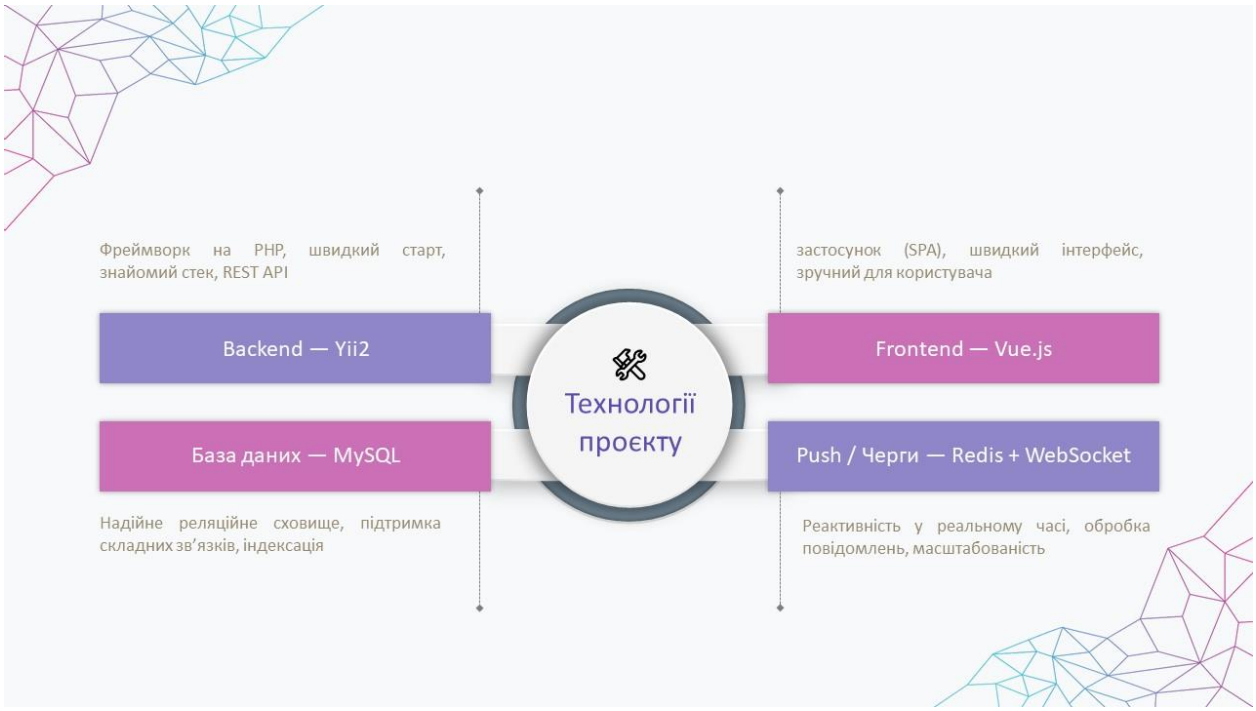
Актуальність теми

Проблема ринку:
Малий бізнес масово переходить на онлайн-продажі, але стикається з труднощами в обробці замовлень - багато процесів залишаються ручними.

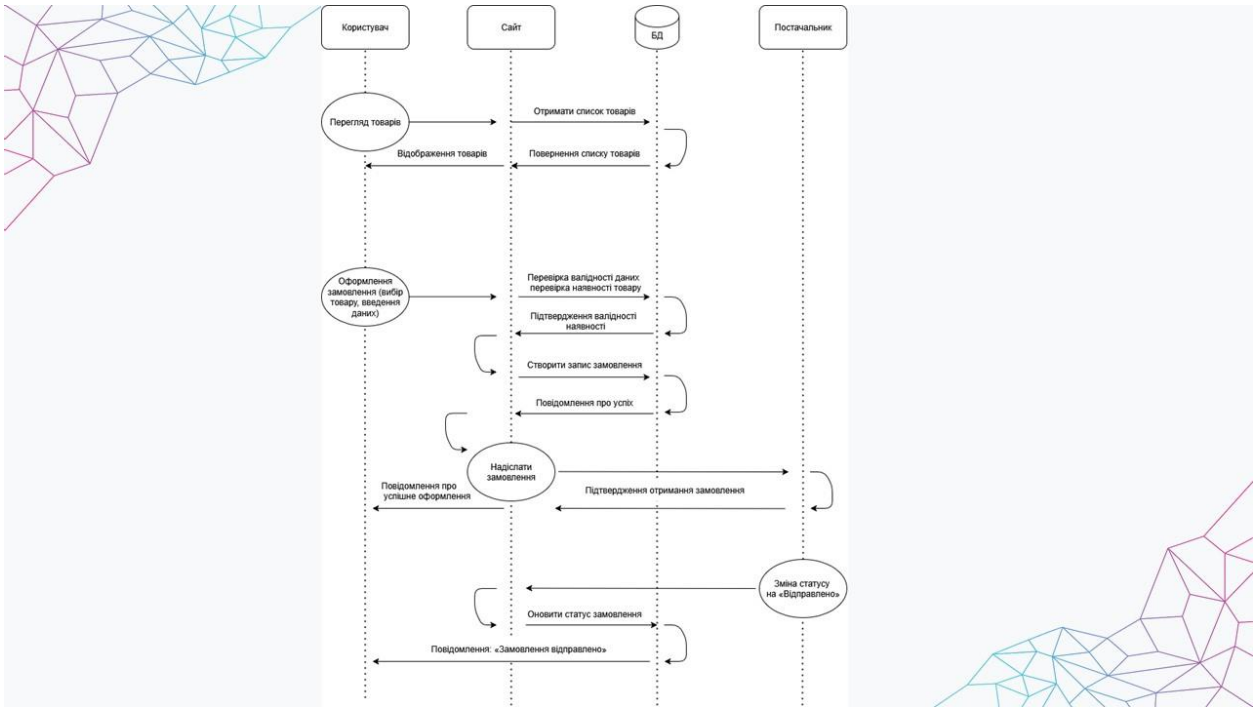
Моє рішення:
Власна автоматизована система, яка адаптована під український бізнес, працює швидко та не вимагає великих витрат.



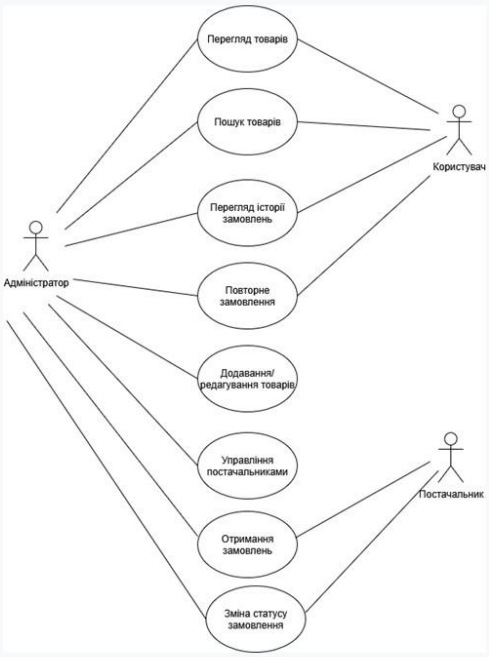
Недоліки готових рішень:
Shopify і WooCommerce або дорогі, або складні в налаштуванні. Вони не враховують потреб українського ринку — доставки, мови, звітності.




Процес оформлення замовлення
01
Вибір товару
Користувач переглядає список товарів, додає потрібне до кошика.
02
Оформлення замовлення
Заповнює контактні дані, обирає спосіб доставки, натискає «Замовити».
03
Обробка системою
Система перевіряє наявність, створює запис у базі, обирає постачальника, встановлює статус.
04
Моніторинг і виконання
Система оновлює статус, всі дії фіксуються. Адміністратор контролює хід виконання та може вносити зміни.

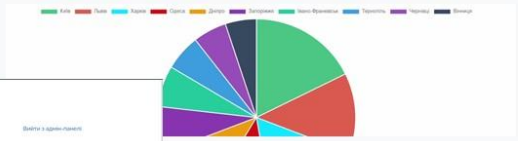

Користувач
Сайт
БД
Постачальник
Перегляд товарів
Отримати список товарів
Відображення товарів
Повернення списку товарів
Оформлення замовлення (вибір товару, введення даних)
Перевірка валідності даних перевірка наявності товару
Підтвердження валідності наявності
Створити запис замовлення
Повідомлення про успіх
Надіслати замовлення
Повідомлення про успішне оформлення
Підтвердження отримання замовлення
Зміна статусу на «Відправлено»
Оновити статус замовлення
Повідомлення: «Замовлення відправлено»

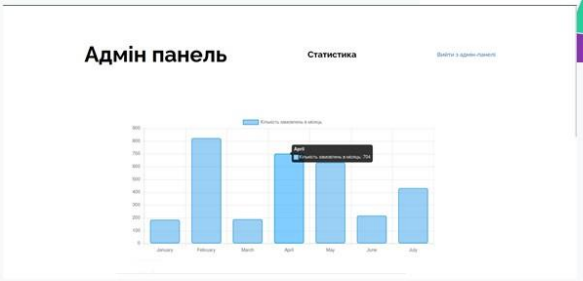
Діаграма прецедентів



Аналітика замовлень

Графіки динаміки та географії продажів для ухвалення бізнес-рішень





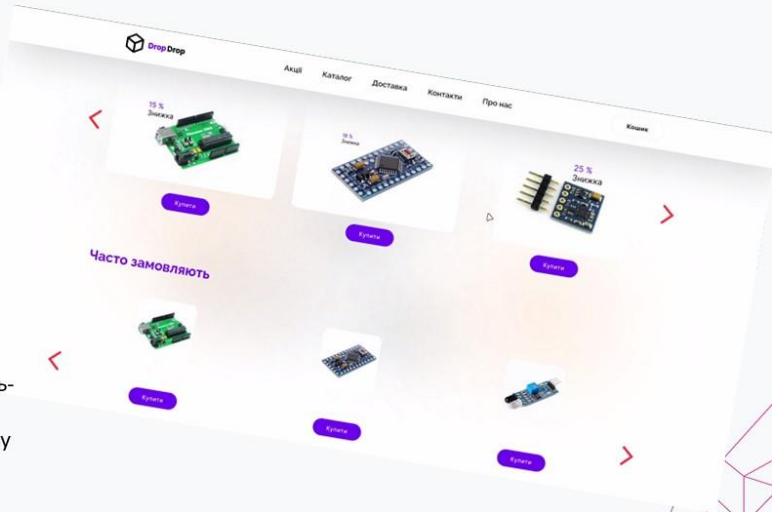
Стимулювання продажів

- **Часто замовляють**

Товари автоматично сортуються за кількістю замовлень. Користувач бачить найпопулярніші позиції — це підвищує довіру та сприяє продажам.

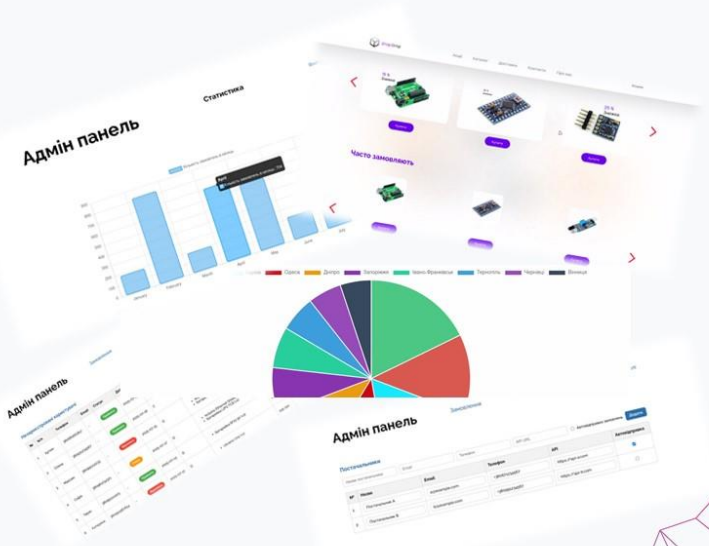
- **Акції**

Адміністратор може додати будь-який товар до розділу «Акції», встановити знижку, і вона одразу відображається на сайті.
Це зручно для розпродажів та маркетингових кампаній.



Висновки

Таким чином, мною реалізовано повноцінну дропшипінг-платформу з усіма базовими функціями: обробка замовлень, аналітика, управління товарами, акціями, клієнтами та постачальниками. Вона легко масштабується, адаптована під українські реалії, і може бути основою для розвитку малого та середнього бізнесу.



№ докумен-та	Позначення	Найменування	Додаткові відомості
		Текстові документи	
1	ГЮИК. 503320.001ВД	Пояснювальна записка	А4, 69 с.
		Додаткові матеріали	
2		Код програми для	А4, 10 с.
		обробки CRUD та	
		оновлення замовлень	
3		Демонстраційний матеріал	А4, 10 с.

Змін.	Арк.	Номер докум.	Підпис	Дата	ГЮИК. 503320.001ВД			
Розроб.		Ноздряков А.А			Розроблення системи автоматизації для управління замовленнями у моделі дропшипінг-бізнесу	Літера	Аркуш	Аркушів
Перевір.		Хрустальова С.В				Н		1
						Кафедра КІТАР ХНУРЕ		
Н.контр.		Стародубцев М. Г.						
Затв.		Невлюдов І.Ш.						