# Sklep Internetowy

Autor – Piotr Makosiej

Jest to aplikacja internetowa, która umożliwia sprzedaż online różnych produktów przez używający ją sklep. Posiada system logowania, gdzie użytkownicy mogą posiadać jedną z trzech ról: klient, pracownik oraz administrator. Klient rejestruje się przy pomocy systemu logowania, pracownika może dodać admin, a konto admina jest tworzone przy tworzeniu bazy danych.

**Klient** może kupować produkty wybierając je z listy. W przypadku braku odpowiedniej ilości produktów w magazynie składane jest zamówienie, które realizowane jest, gdy tylko w magazynie pojawi się wystarczająca ilość produktów.

**Pracownik** ma dodatkowe funkcjonalności w porównaniu do Klienta. Może on dodawać kategorię, dostawców oraz produkty. Każdy produkt ma swojego dostawce i kategorię.

**Admin** posiada wszystkie możliwości co Pracownik oraz może dodawać pracowników.

# OPIS BAZY DANYCH

## Widoki:

- products_view (Jest używany na głównej stronie do wyświetlania najważniejszych informacji o produktach)
- categories_view (Jest używany do wyświetlania listy kategorii przy dodawaniu produktu)
- suppliers_view (Jest używany do wyświetlania listy dostawców przy dodawaniu produktu)
- not_realized_orders (Jest używany w obiekcie klasy Observer, zwraca zamówienia, które nie zostały zrealizowane)

## Funkcje:

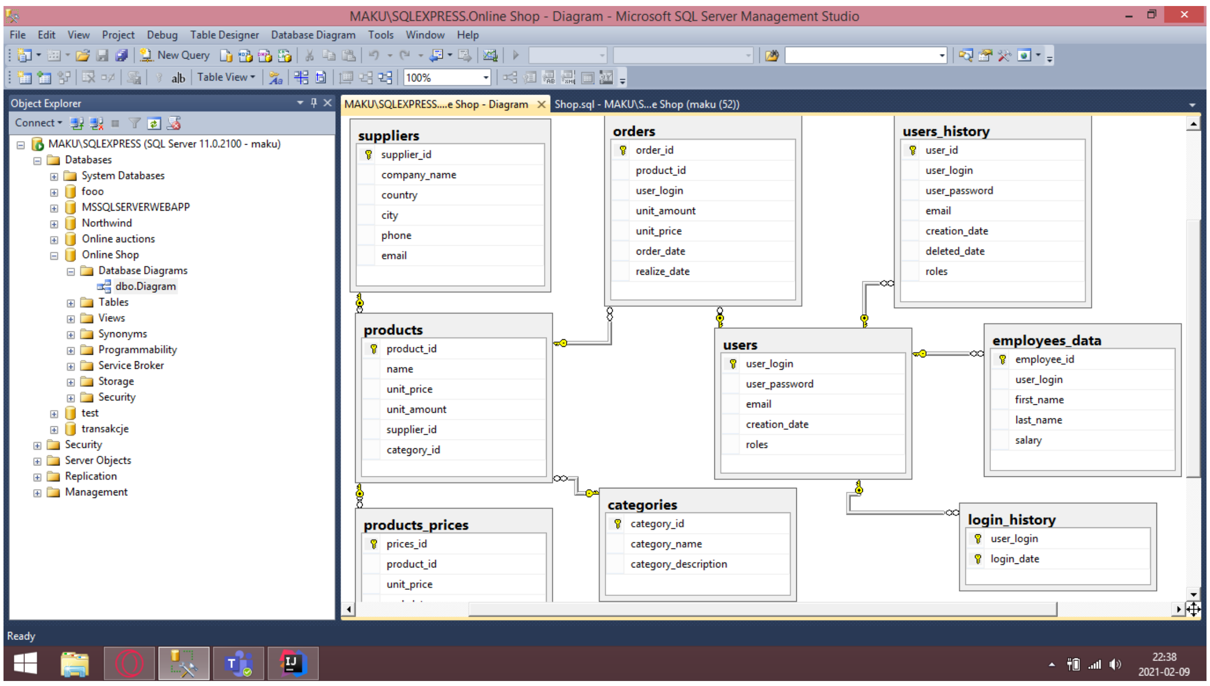
- function_add_user (Sprawdza czy można zarejestrować użytkownika)
- signIn (Sprawdza poprawność danych przy logowaniu)
- show_role (Zwraca rolę użytkownika o podanym loginie)
- product_view_with_details (Zwraca szczegółowe dane wybranego produktu)
- before_add_order (Sprawdza czy jest wystarczająca ilość produktów do sprzedaży)
- before_realize_order (Sprawdza czy zamówienie może być zrealizowane)

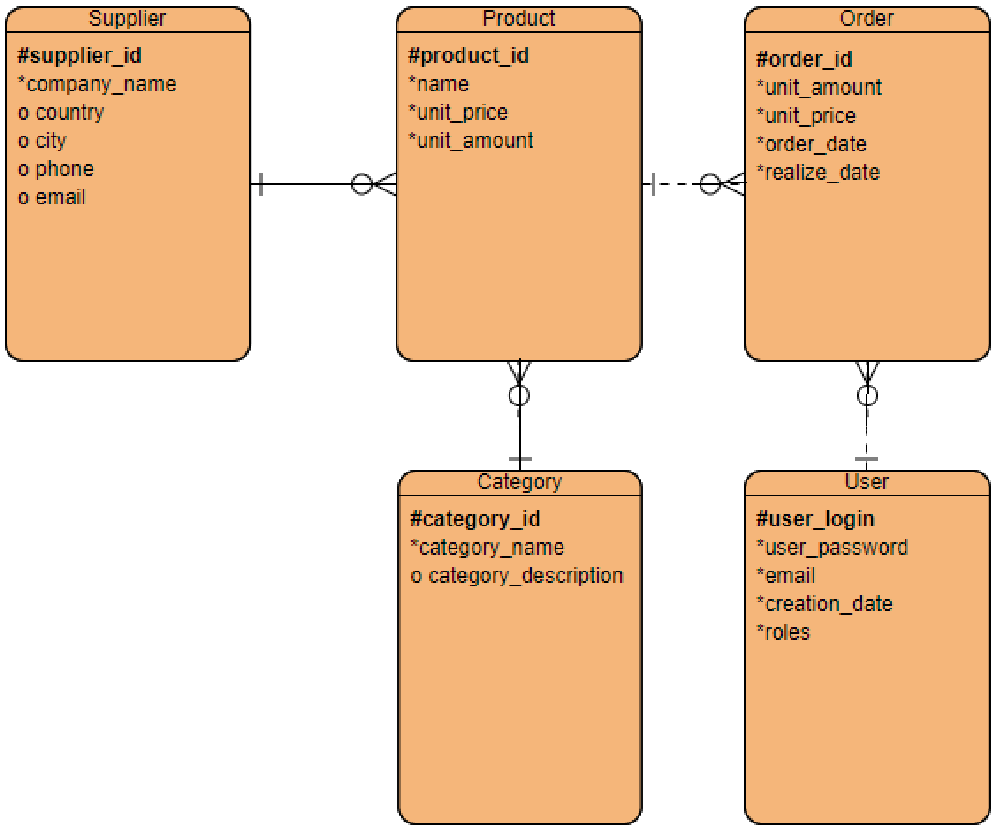
## Procedury:

- add_user (Dodaje nowego użytkownika)
- sign_id (Zapisuje logowanie)
- add_product (Dodaje nowy produkt lub uzupełnia jego ilość. Może również zmieniać cenę)
- add_supplier (Dodaje nowego dostawce lub go aktualizuje)
- add_category (Dodaje nową kategorię lub ją aktualizuje)
- add_employee (Dodaje nowego pracownika)
- add_order (Realizuje zakup produktu lub tworzy oczekujące zamówienie)
- realize_order (Realizuje oczekujące zamówienie)

## Wyzwalacze:

- delete_user (Przenosi usuniętego użytkownika do historycznej tabeli)
- Change_unit_price (Wprowadza dane do tabelki z historycznymi cenami)


MAKU\SQLEXPRESS.Online Shop - Diagram - Microsoft SQL Server Management Studio
File Edit View Project Debug Table Designer Database Diagram Tools Window Help
Object Explorer
MAKU\SQLEXPRESS (SQL Server 11.0.2100 - maku)
Databases
System Databases
fooo
MSSQLSERVERWEBAPP
Northwind
Online auctions
Online Shop
Database Diagrams
dbo.Diagram
Tables
Views
Synonyms
Programmability
Service Broker
Storage
Security
test
transakcje
Security
Server Objects
Replication
Management
MAKU\SQLEXPRESS....e Shop - Diagram
Shop.sql - MAKU\S...e Shop (maku (52))
suppliers
supplier_id
company_name
country
city
phone
email
orders
order_id
product_id
user_login
unit_amount
unit_price
order_date
realize_date
users_history
user_id
user_login
user_password
email
creation_date
deleted_date
roles
products
product_id
name
unit_price
unit_amount
supplier_id
category_id
users
user_login
user_password
email
creation_date
roles
employees_data
employee_id
user_login
first_name
last_name
salary
products_prices
prices_id
product_id
unit_price
categories
category_id
category_name
category_description
login_history
user_login
login_date
Ready
22:38
2021-02-09


Supplier
#supplier_id
*company_name
o country
o city
o phone
o email
Product
#product_id
*name
*unit_price
*unit_amount
Order
#order_id
*unit_amount
*unit_price
*order_date
*realize_date
Category
#category_id
*category_name
o category_description
User
#user_login
*user_password
*email
*creation_date
*roles

# Skrypt:

```sql
if OBJECT_ID('login_history') is not null drop table login_history
if OBJECT_ID('employees_data') is not null drop table employees_data
if OBJECT_ID('orders') is not null drop table orders
if OBJECT_ID('users_history') is not null drop table users_history
if OBJECT_ID('products') is not null drop table products
if OBJECT_ID('categories') is not null drop table categories
if OBJECT_ID('suppliers') is not null drop table suppliers
if OBJECT_ID('users') is not null drop table users
if OBJECT_ID('products_prices') is not null drop table products_prices

-- Tworzenie tabel
go
Create table users(
user_login nvarchar(20) primary key,
user_password nvarchar(20) not null,
email nvarchar(50) unique not null,
creation_date date not null,
roles int not null
)

go
create table login_history(
user_login nvarchar(20) references users(user_login),
login_date datetime,
primary key (user_login,login_date)
)

go
Create table employees_data(
employee_id int primary key identity(1,1),
user_login nvarchar(20) references users(user_login),
first_name nvarchar(20),
last_name nvarchar(20),
salary int
)

go
create table categories(
category_id int primary key identity(1,1),
category_name nvarchar(50) not null,
category_description nvarchar(200) null
)

go
create table suppliers(
supplier_id int primary key identity(1,1),
company_name nvarchar(40) not null,
country nvarchar(40) null,
city nvarchar(40) null,
phone nvarchar(40) null,
email nvarchar(40) null
)

go
create table products(
product_id int primary key identity(1,1),
name Nvarchar(40),
unit_price int,
unit_amount int,
supplier_id int references suppliers(supplier_id),
category_id int references categories(category_id)
)

go
create table orders(
order_id int primary key identity(1,1),
product_id int references products(product_id),
user_login nvarchar(20) references users(user_login),
unit_amount int,
unit_price int,
order_date date,
realize_date date)

-- CHECK
ALTER TABLE orders
ADD CONSTRAINT orders_check_dates CHECK (order_date<=realize_date)

go
Create table users_history(
user_id int primary key identity(1,1),
user_login nvarchar(20) references users(user_login),
user_password nvarchar(20),
email nvarchar(50) unique,
creation_date date,
deleted_date date,
roles int not null
)

go
create table products_prices(
prices_id int primary key identity(1,1),
product_id int references products(product_id),
unit_price int not null,
end_date datetime
)

-- Koniec tworzenia tabel

--admin
go
insert into users values
('maku','123','p_makos@example.com',GETDATE(),2)

-- Tworzenie widoków

go
if OBJECT_ID('products_view') is not null drop view products_view
go
create view products_view as
select name,unit_price,category_name,unit_amount,product_id from products as P
join categories as C on C.category_id=P.category_id

go
if OBJECT_ID('categories_view') is not null drop view categories_view
go
create view categories_view
as select category_name as name,category_id from categories

go
if OBJECT_ID('suppliers_view') is not null drop view suppliers_view
go
```

```sql
create view suppliers_view
as select company_name as name,supplier_id from suppliers

go
if OBJECT_ID('not_realized_orders') is not null drop view not_realized_orders
go
create view not_realized_orders as
select user_login,products.name,products.product_id,order_id from orders
join products on products.product_id=orders.product_id
where realize_date is null

-- Koniec tworzenia widoków

-- Tworzenie funkcji

go
if OBJECT_ID('function_add_user') is not null drop function function_add_user
go
create function function_add_user(
@login nvarchar(20),
@password nvarchar(20),
@email nvarchar(50)
)
returns int
as
begin
if (select user_login from users where user_login=@login) is not null return 1;
else if (select email from users where email=@email) is not null return 2;
return 0;
end


go
if OBJECT_ID('signin') is not null drop function signin
go
create function signin(
@login nvarchar(20),
@password nvarchar(20)
)
returns int
as
begin
if (select user_login from users where user_login=@login and user_password=@password)
is not null return 1;
else if (select user_login from users where user_login=@login) is not null return 2;
return 0;
end


go
if OBJECT_ID('show_role') is not null drop function show_role
go
create function show_role(
@login nvarchar(20)
)
returns int
as
begin
if (select user_login from users where user_login=@login) is not null
       begin
       return (select roles from users where user_login=@login);
       end
```

```sql
return -1;
end


go
if OBJECT_ID('product_view_with_details') is not null drop function
product_view_with_details
go
create function product_view_with_details(@product_id int)
returns table as
return
(select
name,unit_price,category_name,category_description,unit_amount,company_name,country,ci
ty, product_id from products as P
join categories as C on C.category_id=P.category_id
join suppliers as S on S.supplier_id=P.supplier_id
where product_id=@product_id)


go
if OBJECT_ID('before_add_order') is not null drop function before_add_order
go
create function before_add_order(
@product_id int,
@unit_amount int
)
returns int
as
begin
if @unit_amount<=(select unit_amount from products where product_id=@product_id)
return 1;
return 0;
end


go
if OBJECT_ID('before_realize_order') is not null drop function before_realize_order
go
create function before_realize_order(@order_id int)
returns int
as
begin
if exists (select * from orders
                    join products on products.product_id=orders.product_id
                    where order_id=@order_id and realize_date is null and
products.unit_amount>=orders.unit_amount)
       return 1;
return 0;
end

-- Koniec Tworzenia funkcji

-- Tworzenie procedur

go
if OBJECT_ID('add_user') is not null drop procedure add_user
go
create procedure add_user(
@login nvarchar(20),
@password nvarchar(20),
@email nvarchar(50)
)
```

```sql
as
begin
insert into users values
(@login,@password,@email,GETDATE(),0);
end

go
if OBJECT_ID('sign_in') is not null drop procedure sign_in
go
create procedure sign_in(
@login nvarchar(20)
)
as
begin
insert into login_history values
(@login,GETDATE());
end

go
if OBJECT_ID('add_product') is not null drop procedure add_product
go
create procedure add_product
(
@product nvarchar(40),
@category nvarchar(50),
@supplier nvarchar(40),
@unitPrice int,
@unitAmount int
)
as
begin
if  exists (select product_id from products
       join suppliers on products.supplier_id=suppliers.supplier_id
       join categories on products.category_id=categories.category_id
       where name=@product and category_name=@category and company_name=@supplier)
       begin
              update products
              set unit_amount=unit_amount+@unitAmount
              where name=@product and category_id=(select category_id from categories
where category_name=@category)
              and supplier_id=(select supplier_id from suppliers where
company_name=@supplier)
              if @unitPrice>0
                     update products
                     set unit_price=@unitPrice
                     where name=@product and category_id=(select category_id from
categories where category_name=@category)
                     and supplier_id=(select supplier_id from suppliers where
company_name=@supplier)
       end else
       begin
       if (select category_name from categories where category_name=@category) is null
       insert into categories (category_name) values
       (@category)
       if (select company_name from suppliers where company_name=@supplier) is null
       insert into suppliers(company_name) values
       (@supplier)
       insert into products values
       (@product,@unitPrice,@unitAmount,(select supplier_id from suppliers where
company_name=@supplier),(select category_id from categories where
category_name=@category))
       end
```

```
end

go
if OBJECT_ID('add_supplier') is not null drop procedure add_supplier
go
create procedure add_supplier
(
@name nvarchar(40),
@country nvarchar(40),
@city nvarchar(40),
@phone nvarchar(40),
@email nvarchar(40)
)
as
begin
if  exists (select company_name from suppliers where company_name=@name)
       begin
              update suppliers
              set country=@country
              where company_name=@name
              update suppliers
              set city=@city
              where company_name=@name
              update suppliers
              set phone=@phone
              where company_name=@name
              update suppliers
              set email=@email
              where company_name=@name
       end else
       begin
       insert into suppliers values
       (@name,@country,@city,@phone,@email)
       end
end

go
if OBJECT_ID('add_category') is not null drop procedure add_category
go
create procedure add_category
(
@name nvarchar(50),
@description nvarchar(200)
)
as
begin
if  exists (select category_name from categories where category_name=@name)
       begin
              update categories
              set @description=@description
              where category_name=@name
       end else
       begin
       insert into categories values
       (@name,@description)
       end
end

go
if OBJECT_ID('add_employee') is not null drop procedure add_employee
go
create procedure add_employee
```

```
(
@login nvarchar(20),
@password nvarchar(20),
@email nvarchar(50),
@firstName nvarchar(20),
@lastName nvarchar(20),
@salary int)
as
begin
if  not exists (select * from users where user_login=@login)
       begin
       insert into users
(user_login,user_password,email,creation_date,roles)values(@login,@password,@email,GET
DATE(),1)
       insert into employees_data values(@login,@firstName,@lastName,@salary)
       end
end

go
if OBJECT_ID('add_order') is not null drop procedure add_order
go
create procedure add_order
(
@product_id int,
@unit_amount int,
@user_login nvarchar(20))
as
begin
declare @amount int;
set @amount = (select unit_amount from products where product_id=@product_id)
if(@unit_amount<=@amount)
       begin
       insert into orders values
       (@product_id,@user_login,@unit_amount,(select unit_price from products where
product_id=@product_id),GETDATE(),GETDATE())
       update products
       set unit_amount=unit_amount-@unit_amount
       where product_id=@product_id
       return 1;
       end
else begin
        insert into orders values
        (@product_id,@user_login,@unit_amount,(select unit_price from products where
product_id=@product_id),GETDATE(),null)
        return 0;
       end
end

go
if OBJECT_ID('realize_order') is not null drop procedure realize_order
go
create procedure realize_order(@order_id int)
as
begin
if exists (select * from orders
                    join products on products.product_id=orders.product_id
                    where order_id=@order_id and realize_date is null and
products.unit_amount>=orders.unit_amount)
       begin
       update orders
       set realize_date=GETDATE()
       where order_id=@order_id
```

```sql
	update products
	set unit_amount-= (select unit_amount from orders where order_id=@order_id)
	where product_id = (select product_id from orders where order_id=@order_id)
	end
return 0;
end

-- Koniec tworzenia procedur

-- Tworzenie wyzwalaczy

go
if OBJECT_ID('delete_user') is not null drop trigger delete_user
go
create trigger delete_user on users
after delete
as declare
	@user_login nvarchar(20),
	@user_password nvarchar(20),
	@email nvarchar(50),
	@creation_date date,
	@deleted_date date,
	@roles int;

	set @user_login = (select user_login from deleted);
	set @user_password  = (select user_password from deleted);
	set @email = (select email from deleted);
	set @creation_date = (select creation_date from deleted);
	set @deleted_date = GETDATE();
	set @roles = (select roles from deleted);
begin
insert into users_history values
(@user_login,@user_password,@email,@creation_date,@deleted_date,@roles)
end


go
if OBJECT_ID('change_unit_price') is not null drop trigger change_unit_price
go
create trigger change_unit_price on products
after update
as
declare
	@product_id int,
	@unit_price int;
set @product_id = (select product_id from deleted);
set @unit_price = (select unit_price from deleted);
if update(unit_price)
begin
insert into products_prices values
(@product_id,@unit_price,GETDATE())
end

-- Koniec tworzenia wyzwalaczy

go
select * from products
select * from categories
select * from suppliers
select * from orders
select * from users
select * from employees_data
select * from login_history
select * from users_history
select * from products_prices
```