# “Ministore” v1.0 Documentation

Created: 12.08.2017  
by: idealdesigns

Thank you for purchasing my template. If you have any questions that are beyond the scope of this help file, please feel free to email via my user page contact form [**here**](https://codecanyon.net/user/idealdesigns). Thanks so much!

This is my first release and more updates and improvements will come. Feel free to contact me if you want additional features or fixes.

Table of Contents

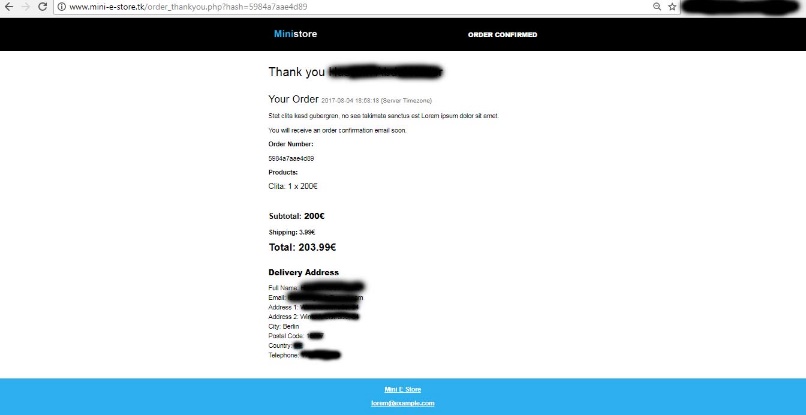
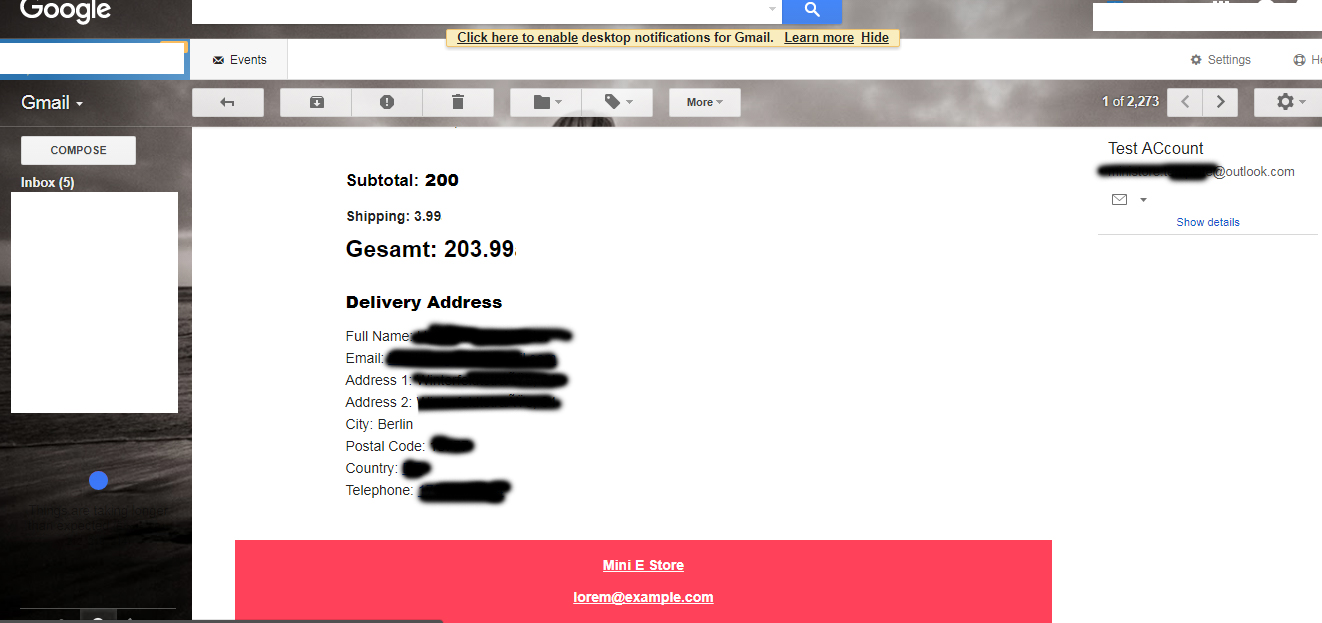
[“Ministore” v1.0 Documentation 1](#_Toc489658741)

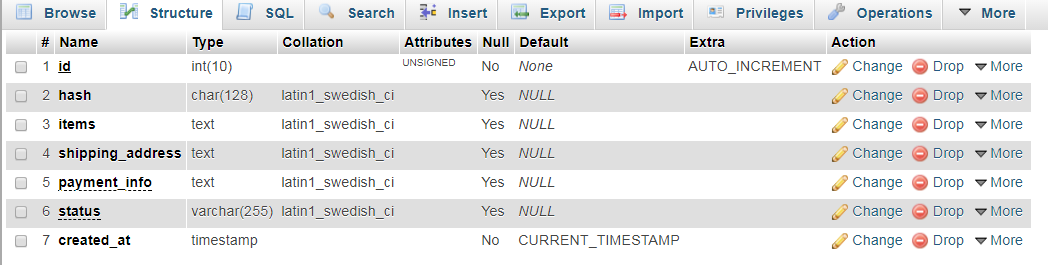
[A) Dynamic features of the Ministore template 3](#_Toc489658742)

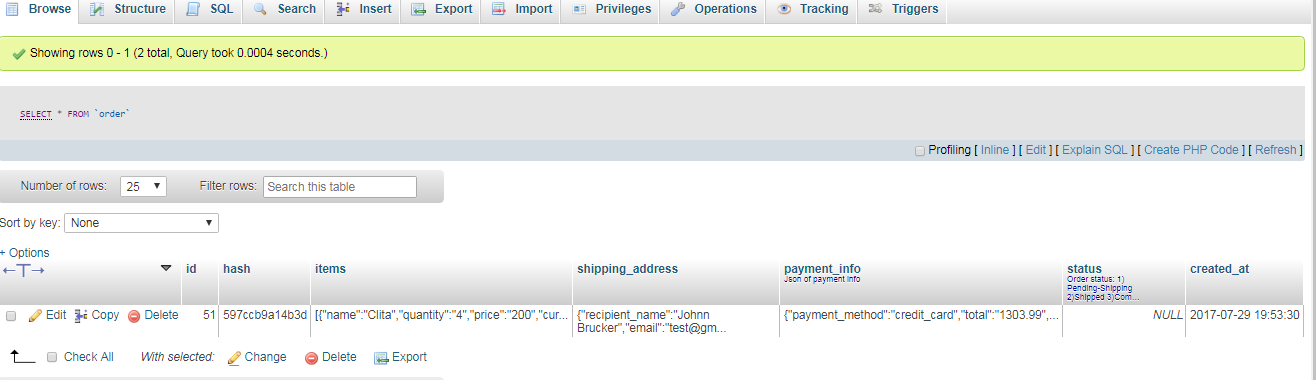
[Installing order.sql 7](#_Toc489658743)

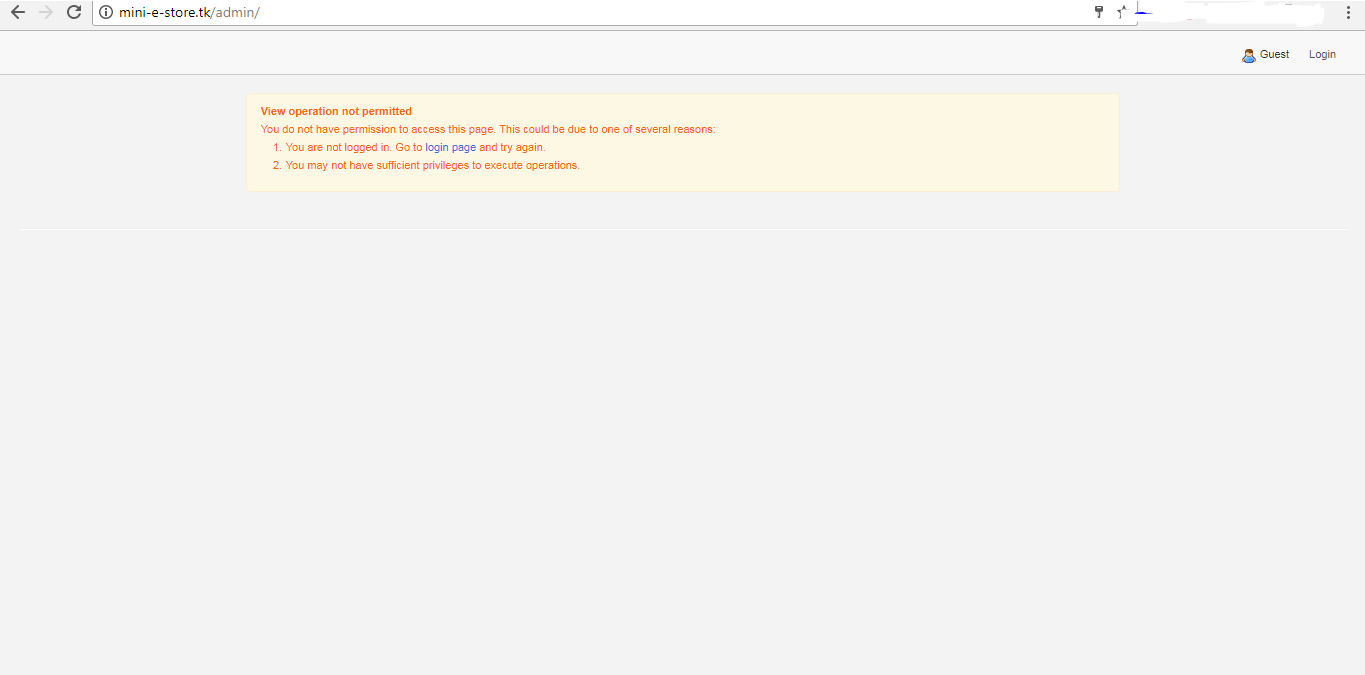
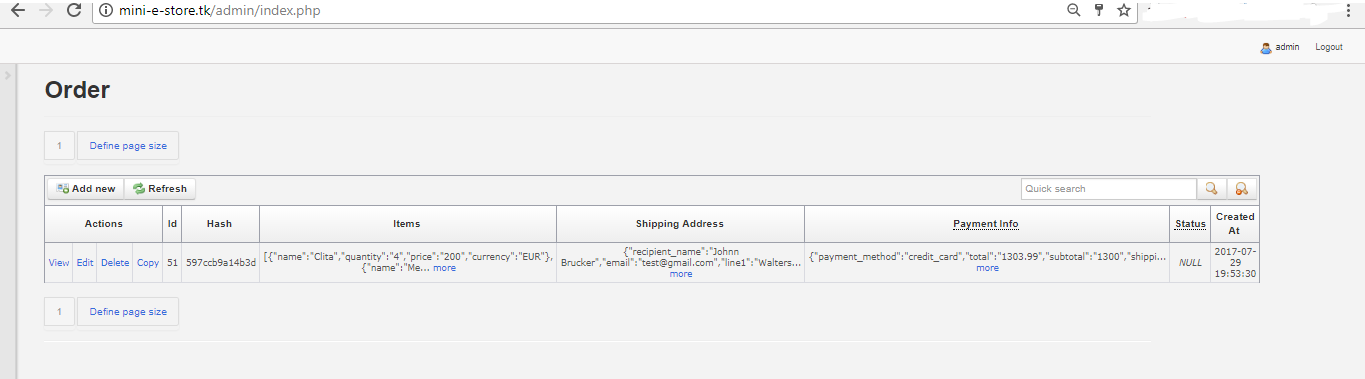
[CHANGE LOG 8](#_Toc489658744)

# A) Dynamic features of the Ministore template

1. **Order form**The prices and total are calculated using AngularJS, the implementation can be found in “*js/main.js”*Don’t forget to modify the prices in index.php also.The submission is done using jQuery Ajax function and the data is submitted to the server.
2. **PayPal integration**  
     
   simplest possible PayPal Integration approach was used.  
   See <https://github.com/paypal/paypal-checkout> for more information  
     
   Do not forget to update the API keys and setting the environment either sandbox or production.  
     
   Files “js/custom.js” and “process\_paypal\_order.php”
3. **Stripe integration**  
   Custom Stripe JavaScript integration was used. For more information:   
   <https://stripe.com/docs/checkout#integration-custom>  
     
   Unlike PayPal, the actual charge will be done on the Server with PHP code. First step you will be given a token in javascript then this Token is used in the PHP code to charge the customer’s credit card.  
     
   For testing you can use Test Mode credit cards from Stripe:  
   <https://stripe.com/docs/testing#cards>  
     
   Do not forget to update the API keys for sandbox or production in “main.js” and “/includes/stripe\_create\_charge.php”.  
     
   Files “js/custom.js” and “process\_stripe\_order.php” and “/includes/stripe\_create\_charge.php”
4. **Order Thank You page**PHP file “order\_thankyou.php”
5. **Sending Order confirmation Email to the Customer and to the Admin**After the purchase an order confirmation email will be sent to the customer’s email and to the admin using PHPMailer library.  
   The PHP files that are responsible for that are “process\_paypal\_order.php” and “process\_stripe\_order.php”.  
     
   Example of customer order confirmation Email:  
   
6. **Record Orders in MySQL Database table “Order”**

The SQL code is executed in the files “process\_paypal\_order.php” and “process\_stripe\_order.php”  
  
phpMyAdmin screenshots to demonstrate:  




1. **Admin area to view and edit the Orders**Go to domain.com/admin  
   The default username is “admin” and the password “000”  
     
   The password and username is hardcoded in“admin/authorization.php”.   
   Change database connection settings in “admin/ phpgen\_settings.php”.  
     
     
   
2. **Sass and Grunt**Sass style is in “sass/custom.scss”.  
   You need to run “grunt” or “grunt watch” commands and it will generate a file called “css/custom.css” when you modify custom.scss.  
     
   If you get an error when running “grunt” command then you can find more information about setting up your development environment for Windows or OS X or Linux here:  
   http://ryanchristiani.com/getting-started-with-grunt-and-sass/  
   and   
   <https://www.lynda.com/Grunt-js-tutorials/Running-Grunt-Sass-first-time/372540/385119-4.html>
3. **Working PHP Contact Form**Contact form PHP code can be found in “message\_sent.php”  
   PHPMailer is used to send the email.

# Installing order.sql

First step create a database then run the SQL code in “order.sql”, this will create the **Order** table in the database correctly because the PHP code relies on this table structure, any change in the SQL code must be synced with the PHP code and vise versa.

--

-- Table structure for table `order`

--

CREATE TABLE IF NOT EXISTS `order` (

`id` int(10) unsigned NOT NULL,

`hash` char(128) DEFAULT NULL,

`items` text,

`shipping\_address` text,

`payment\_info` text COMMENT 'Json of payment info',

`status` varchar(255) DEFAULT NULL COMMENT 'Order status: 1) Pending-Shipping 2)Shipped 3)Completed',

`created\_at` timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP

) ENGINE=InnoDB AUTO\_INCREMENT=52 DEFAULT CHARSET=latin1;

--

-- Indexes for table `order`

--

ALTER TABLE `order`

ADD PRIMARY KEY (`id`), ADD UNIQUE KEY `hash` (`hash`);

--

-- AUTO\_INCREMENT for table `order`

--

ALTER TABLE `order`

MODIFY `id` int(10) unsigned NOT NULL AUTO\_INCREMENT,AUTO\_INCREMENT=52;

The SQL file contains also sample data to be inserted.

# CHANGE LOG

See the file “CHANGELOG.md”