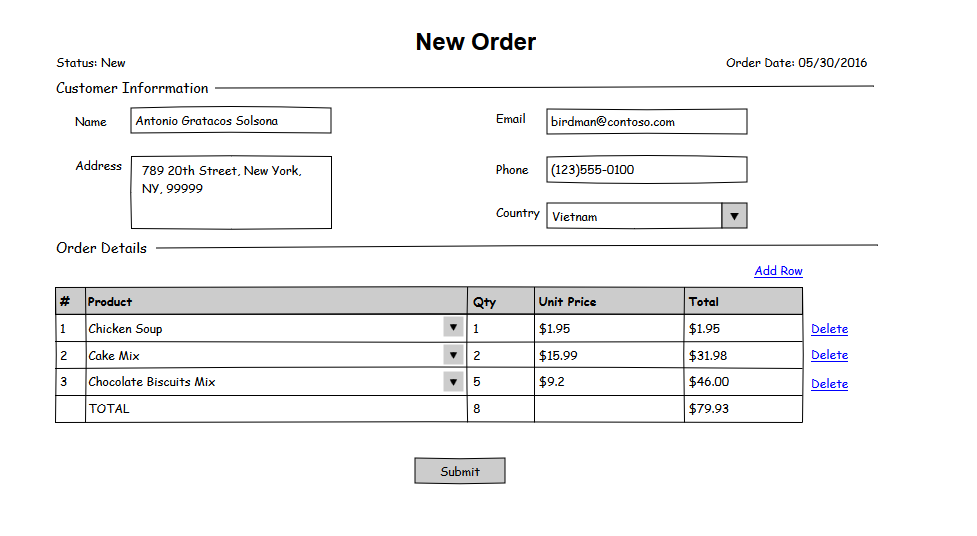
# Functional Requirement:

## New Order:



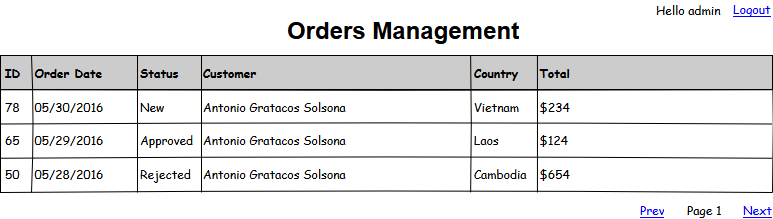
### UI Validation:

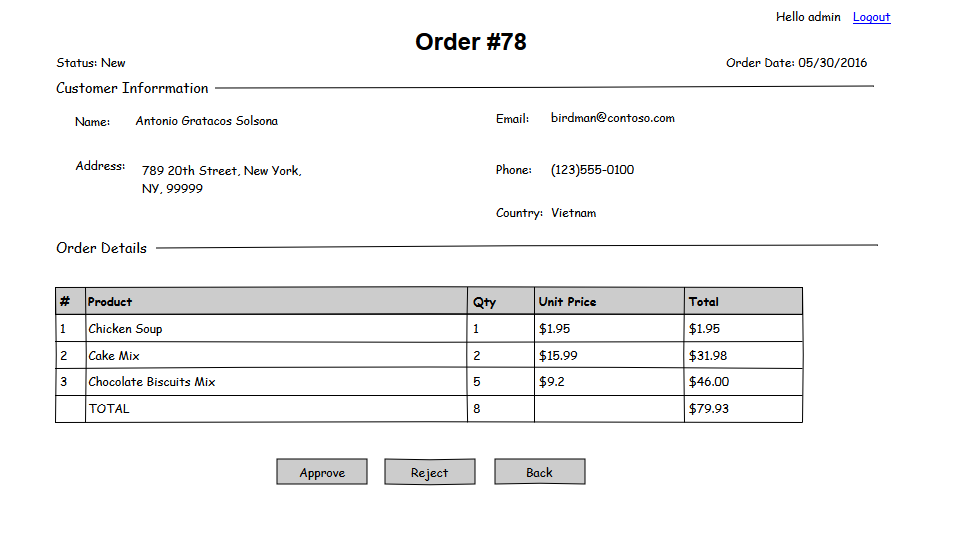
* Field Name is a required field
* Email should be validated & required field

### Business Logic:

* Customer Information data will be saved into table Customers
* Order Details data will saved into table OrderDetails
* Order information also hold in table Order. Table Order/OrderDetails is master/detail (1-M) relationship
* Click 'Add Row' link will add a row to grid
* Click 'Delete' link should remove the selected row
* Total column of each row is = Qty \* Unit Price It should be calculated instantly
* The product list for selection (combo box) will loaded from table Products
* When the user selects a product from the list then 'Unit Price' will be filled automatically for that row
* The TOTAL value is sum of Total column. This value should be updated instantly when there is any change to order detail information
* The default status of Order is New
* Order Date will be filled automatically with current date
* Click Submit button to send the order

## Orders Management:



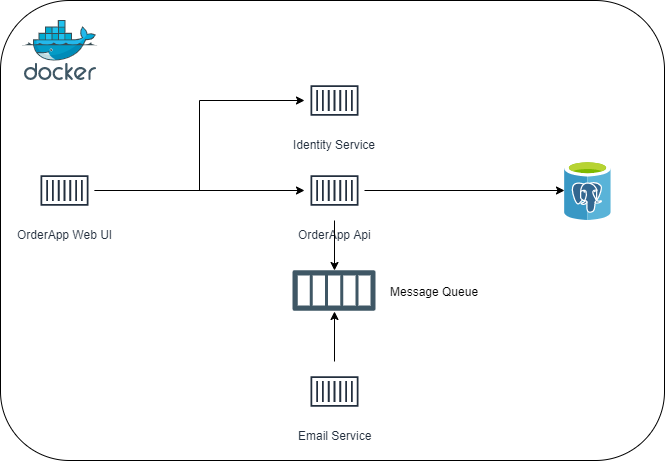


### Business Logic:

* The grid shows the latest orders first (1st image)
* Data must be paged. The number of records per page is 25
* Column 'Order Date' & 'Customer' should be sortable
* The 'Customer' column of row is a hyperlink. When the user clicks on it then it redirects to view order page (see the 2nd image)
* All data in view order page is read-only. If the order status is New then the admin can approve or reject the selected order. If the admin click 'Approve' button then the status of order is Approved. In case the admin selects 'Reject' button , the status should be Rejected
* When the order status is Approved or Rejected then both buttons (Approve & Reject) are hidden
* Click Back button to back to the grid
* Click Logout link to sign out the admin page

# Technical Requirement:

* Application was designed following Microservice Architect and deployed in Docker Container.



### Description:

* **OrderApp Web UI**: User interface that help user interact to system.
* **Technology**: VueJS, Bootstrap 4, nginx
* **Identity Service**: Authorization Server (Just build simple using service in-memory)
* **Technology**: .Net core 2.2, Identity Server 4
* **OrderApp Api**: The main business logic, provide Apis for UI, work CRUD with DB and also apply message queue to communicate to other services.
* **Technology**: .Net core 2.2, gRPC client, Masstransit, AutoMapper, Swagger, Entity Framework Core.
* **Email Service**: Send notify email to client’s email (Just build simple, only write log send to address).
* **Technology**: .Net Core 3.0, gRPC server.
* **Database**: Postgres
* **Message broker**: RabbitMQ