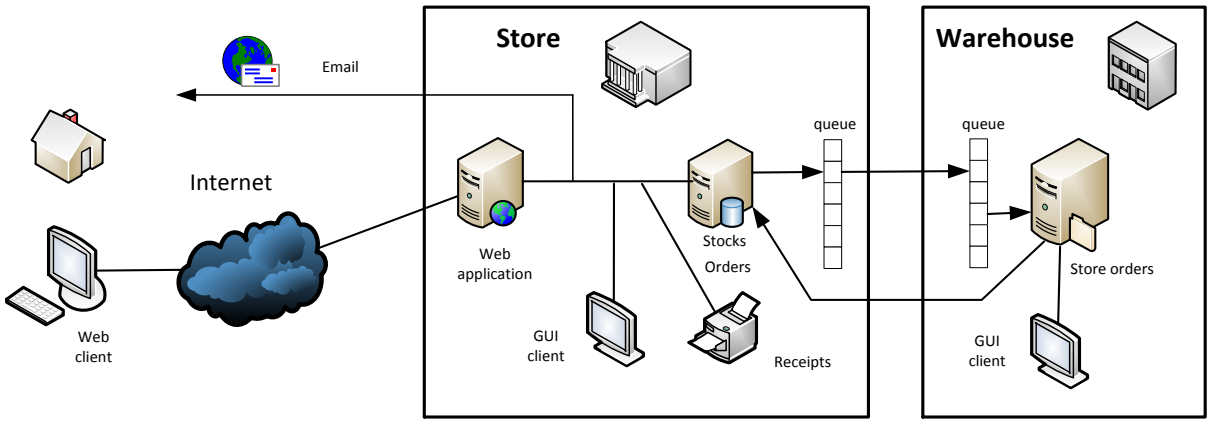
# Distribution and Integration Technologies (TDIN)

Assignment # 2
An enterprise distributed system
2014-15

**Scenario**

A book editor prints and sells book titles (assume only a few titles) and intends to develop a system for coordinating its sells, orders and stock management. The editor owns two facilities physically different, the store with a public and exposition area, and a warehouse for storing larger quantities of books. The editor intends also to make available a web application for remote ordering.

In the store there is a server, hosting the web application, and a persistent record of available titles, price and existences in the store and warehouse. This store server should be a web server (always on) and is connected to the internet. Between the store and warehouse there is also a network connection, but the warehouse computers usually run only in labor hours, including its server.




In the store and warehouse there are also GUI clients operated by the workers of those places.

The main requirements and operation of the all system are as follows:

- The store web server accepts orders from the internet, using a web application. Each order should specify a title (for simplicity assume only a title per order/sell), quantity, client name, address and email. Each order has also a unique identifier. All created orders are stored on the server and have a state. For simplicity we can ignore the payment process in this scenario.
- The store GUI application allows a sell of titles existing on the store for a client visiting the store, or the creation of an order (identical to a web order) if the title only exists on the warehouse. A sell specifies also the client name, book title, quantity and total price, updates the store existence and prints a receipt (use a separate application (console or GUI) to simulate the printer). When a title doesn't exist in the store an order should be created.
- Orders (originating from the web or the GUI store application) can be in one of three states: "waiting expedition", "dispatched at … (date)", "dispatch will occur at … (date)".
- All new orders are recorded in the store server. This server verifies the store stock. If

the stock is enough, the server updates it and sends an email to the client with the details (title, quantity, total price and state). This email replaces the receipt. The state of the order should now be "dispatched at … (next day date)". If the stock is not enough a request in the form of a message (it can simulate a call) is sent to the warehouse server, for a quantity 10 times the initial order volume. In this case the initial order state should be "waiting expedition".

- The warehouse server receives these messages asynchronously through a message queue (which can be linked to a service) and persists its data for ulterior consultation and processing.
- The requests to the warehouse can be consulted and manipulated by a GUI client in the warehouse. When the warehouse is about to ship the title to the store (indicated by an employee in the warehouse GUI client), the store server should be notified (since the store server is always on, this could be a simple remote call) and the order state (in the store) should change to "dispatch should occur at … (today plus 2 days date)". The title will be now physically transported to the store. Assume that the warehouse has always the number of books requested.
- When the store receives the requested books from the warehouse, a clerk must update the store stock and all pending orders that can be satisfied should change state (to "dispatched at ... (today's date)") and a new email should be sent to the client.
- Remote clients, through the web application, can consult the state of their orders.

**Technologies**

Implement this distributed application following the Service Oriented Architecture (SOA) principles. You can choose any appropriate technologies.

The service at the store server should support several operations like:

- consult the store stock of a title
- make a sell (updates the store stock and prints a receipt)
- create an order
- change the state of an order
- any other convenient operations

The service at the warehouse should support operations like:

- get open requests that arrived from the store
- complete a request
- any other convenient operations

The information needed to the servers in the application can be persisted in files or local databases.

**Realization**

The user interface should allow the specified requirements verification in a comfortable way. For testing and demonstration all services, servers and applications can run on the same computer. You can add any functionality considered useful or relevant.

The satisfaction of all requirements, ease of use, other operation functionalities, good practices, are factors taken into consideration for grading.

**Report**

You should write a small report containing a detailed architecture specification (functionality, the modules and their interaction), any testing done, and a graphical representation (screen captures) of the main flows of use.

You should state also all the conditions and instructions to build and run your applications.