Design Document for “Order and Delivery management system” project

Group#08

Rassim Batarayev: %marks

Daniel Khassanov: %marks

Amantur Amangeldiyev: %marks

Dastan Koktalov: % marks

Version	Date	Author	Change
0.1	19/03/14	SM	Initial Document

Table of Contents

1 Introduction

1.1 PURPOSE

The purpose of this document is to explain the design and architecture of the “Order and delivery management system”

1.2 SCOPE

This document covers system decomposition, interfaces, and dependencies, as well as design rationale

1.3 DEFINITIONS, ACRONYMS, ABBREVIATIONS

Term	Description

1.4 DESIGN GOALS (DKO)

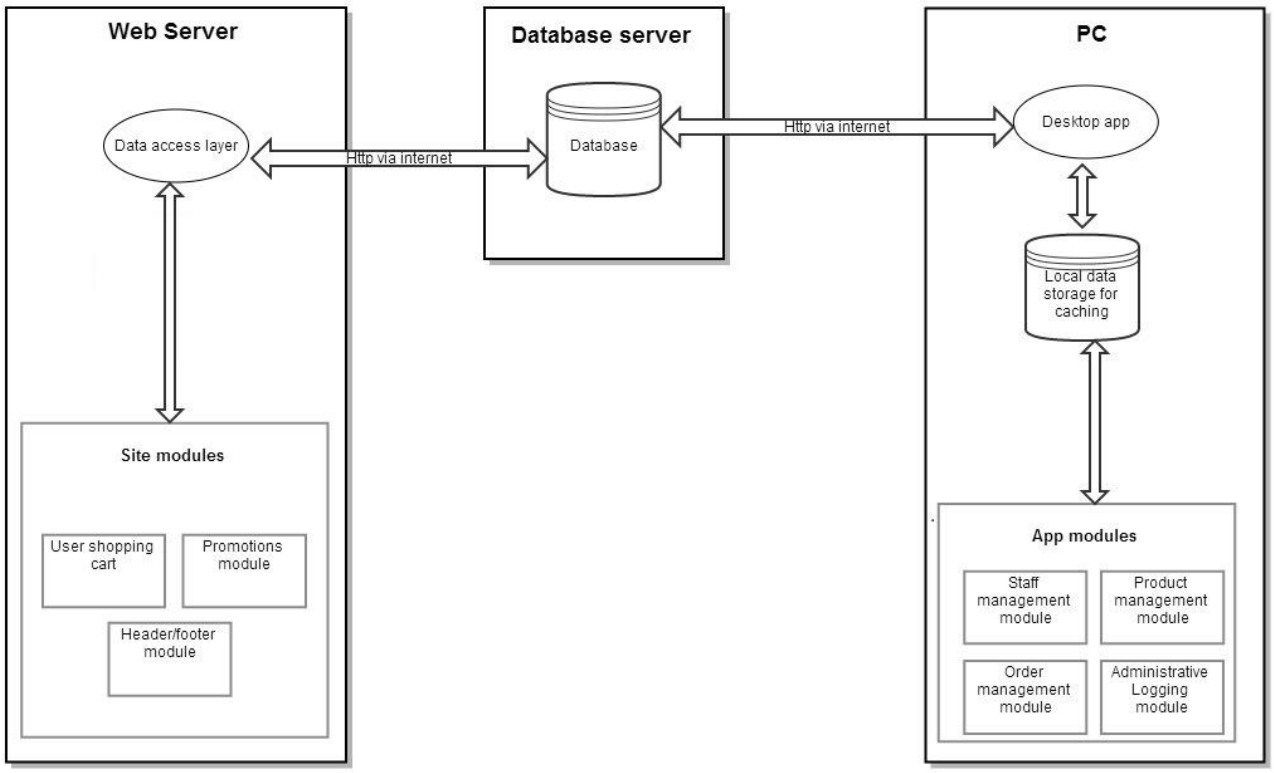
1. Reliability: The core process (major use-cases) must continually function consistently, and loss of any data must be prevented.
2. Maintainability: Our code must be well organized with consistent syntax and relavent naming techniques.
3. Extensibility: The system must facilitate easy extension of the website and application.
4. Response Time: All queries in application and website must give response in 30 seconds including library and database processing time.

2 References

[NONE]

3 Decomposition Description

3.1 MODULE DECOMPOSITION



This project contains desktop application and web site which are connected via database. And consist of 3 layers. Our main (controlling) part is desktop application and it's located in PC layer. Our client side has only web page which has site modules and located in Web Server layer. Last layer is Database server layer. It's bounding layer of other two layers. And it has high priority as desktop application, because it is connects our client side and administrative side.

3.1.1 Staff management module (DKO)

This module is responsible for staff management via desktop application. In this module responsible for managing of staff data. This module has to:

1) Get all data from administration, divide it to small pieces and send the pieces to database.

2) Get data from data cache and show it to administration.

3) By administration’s desire has to delete all data about some staff from database’s staff table’s row by id.

4) By administration’s desire has to edit data’s about some staff from database’s staff table’s row by id.

3.1.2 Product management module (DKH)

This module is responsible for product managing in web site via desktop application. By this module our administration can easily change content of site without code knowledge or other extra knowledge in IT. This module has to:

1) Get all necessary data from data cache, parse it and show in easy GUI form.

2) Get all necessary data from administration that they desire to add, parse it and send it to database.

3) Get all necessary data from administration that they desire to edit, parse it and send it to database.

4) Get all necessary id of product from administration that they desire to delete and send that id to delete proper product from database.

3.1.3 Order management module (AA)

This module is responsible for every order that was come from users via desktop application. By this module our administration can manage orders. This module has to:

1) Get all necessary data from data cache, parse it and show in easy GUI form.

2) Track administrations choice of approve or denial to change status of chosen orders id from database.

3.1.4 Administrative logging module (RB)

This module is responsible for orders logging only for administrator via desktop application. By this module administrator can see all order interactions made by managers. This module has to:

1) Get all necessary data from data cache and show it in easy GUI form.

3.1.5 Session module (AA)

This module is responsible for sessions in our desktop application. This module has to:

1) Get authentication data and validate them before sending to database.

2) Make session for different users and privileges. For example: at the same time there might be sitting administrator and two managers, and they have to make their own deeds without interrupts or privilege changes.

3.2 CONCURRENT PROCESS

3.2.1 Data caching process Description (DKH)

Data caching process is process, which caches all data, which comes from and to desktop application. This process is responsible for data integrity. All queries and data which will be sent to database server, firstly will be checked by this process, and only after checking will be sent to database server.

3.2.2 Data parsing process Description (RB)

Data parsing process is process, which will translate all data from database format to format which will be understandable by web site.

3.3 DATA DECOMPOSITION

3.3.1 <Class 1> Description

3.3.2 <Class 2> Description

3.4 STATES

3.4.1 <State/System 1 > Description

3.4.2 <State/System 2> Description

4 Dependency Description

4.1 INTERMODULE DEPENDENCIES

4.2 INTERPROCESS DEPENDENCIES

4.3 DATA DEPENDENCIES

5 Interface Description

5.1 MODULE INTERFACE

5.1.1 Desktop application code interface

5.1.1.1 public function getStaffList(){

//No parameters

//It is used to get list of all staff and show it in our "Staff" window

//returns parsed staff list, which will be taken by application's staff table

}

5.1.1.2 public function createStaff(email:String, password:String, position:String, photo:QByteArray, fullname: String){

//This function is used to create new staff in our desktop app

//Parameters:

//email - it is email of person. Will be used to login into application, also on this email wil be sent new password after resetting

// password - this is password of user, which will be used to login into application

// position - in which position this person in company (courier/manager)

// photo - this will be photo of a person, which wiil be sent in QByteArray format, and stored in BLOB format of MySQL

// fullname - First and Second name of persons

//return type - Boolean, shows if that query was success or failed

}

5.1.1.3 public function editStaff(email:String){

// This function gets all info of selected person from database by email and gives us ability to change that info

// after changing, all info with changes is sent back to database

// Parameters:

// email - email of person, whose info must be changed

// return type - Boolean, shows if that query was success or failed

}

5.1.1.4 public function deleteStaff(email:String)

// This function deletes from database current person's record

// Parameter:

// email - email of person, who is gone from company

// return type - Boolean, shows if query was success or failed

5.1.1.5 public function login(email:String, password:String){

// This function authenticates user, who wants to enter the application

// Parameters:

// email - email of person, who wants to log in

// password - password of person, who wants to log in

// return type - Boolean, shows if user was logged in succesfully or not

}

5.1.1.6 public function forgotPass(email:String){

// This function sends email of person, who forgot his password to webserver, where password will reset, and new one will be sent to person's email

// Parameter:

// email - email of person, who wants to reset his password

//return type - Boolean, shows if operatio completed succesfully or not

}

5.1.1.7 public function changePass(email:String, oldPassword:String, newPassword:String) {

//This function changes password of person, who wants to change his password to new one

// Parameters:

// email - email of person, who wants to change his password

// oldPassword - old password of person, who wants to change his password

// newPassword - new password of person, to which he wants to change his old password

// return type - Boolean, shows if password was changed succesfully or not

}

5.1.1.8 public function addContent(type:String, name: String, price: int, ingredients: String, photo:QByteArray){

//This function adds new product's record to database with given charachteristics

//Parameters:

// type - which product will be added (pizza, sushi, drinks and etc.)

// name - name of product

// price - price of new product

// ingredients - composition of new product

// photo - photo of new product

}

5.1.1.9 public function editContent(name:String){

// This function gives ability to change products info

// Parameter:

// name - name of product (this is Unique field in our Database)

// return type - Boolean, which shows if operation was successfull or not

}

5.1.1.10 public function deleteContent(name:String){

// This function deletes product's record from our database

// Parameter:

// name - name of product, which will be deleted from database

// return type - Boolean, which shows if operation was successfull or not

}

5.1.1.11 public function addPromo(name:String, desciption:String, photo:QByteArray){

//This function adds promo actions

// Parameters:

// name - name of promo action

// description - full description of promo

// return type - Boolean, which shows if operation was successfull or not

}

5.1.1.12 public function editPromo(name:String){

// This function gives ability to change promo

// Parameter:

// name - name of promo, which must be changed

// return type - Boolean, which shows if operation was successfull or not

}

5.1.1.13 public function deletePromo(name:String){

// This function deletes promo's record from our database

// Parameter:

// name - name of promo, which will be deleted from database

// return type - Boolean, which shows if operation was successfull or not

}

5.1.1.14 public function addGalery(name:String, description:String, photo:QByteArray){

//This function adds news for gallery

// Parameters:

// name - name of news in gallery

// description - full description of news

// photo - photo which will be shown in gallery on main page

// return type - Boolean, which shows if operation was successfull or not

}

5.1.1.15)public function editGallery(name:String){

// This function gives ability to change news in Gallery

// Parameter:

// name - name of news, which must be changed

// return type - Boolean, which shows if operation was successfull or not

}

5.1.1.16 public function deleteGallery(name:String){

// This function gives ability to delete news record from our database

// Parameter:

// name - name of news, which must be deleted

// return type - Boolean, which shows if operation was successfull or not

}

5.1.1.17 public function getOrders(){

// This function gets all active orders from our database, parses and sends to table

//no parameters

//returns parsed orders list, which will be taken by application's orders table

}

5.1.1.18 public function assignCourierToOrder(idCourier:int, idOrder:String){

//This function assigns order to couriers

//Parameters:

//idCourier - id of courier, to whom will be added order

// idOrder - id of order, which must be assigned to courier

//return type - Boolean, which shows if operation was successfull or not

}

5.1.1.19 public function getCourierList(){

// This function will get all courier list from database and show it in "courier assign" window

// No parameters

//returns parsed couriers list, which will be showed in "courier assign" window

}

5.1.2 Site code interface

5.1.2.1 Gallery(blob pic, string desc)

Is used to show gallery and parse data from database.

Parameters:

Pic = the picture that have to be rotated.

Desc = the description text that have to be shown with picture.

Return type void

5.1.2.2 Promo(blob pic ,string name, string desc)

Is used to show promotions and parse data from database.

Parameters:

Pic = the picture that have to be shown in promotions.

Name = the short text that have to be shown always.

Desc = the full description of promotion

Return type void

5.1.2.3 Recommended(blob pic, string name, int price)

Is used to show recommended product in main page and parse data from database.

Parameters:

Pic = the picture of product.

Name = the name of product.

Price = the price of product.

Return type void

5.1.2.4 Menu(blob pic, string name, int price, string desc)

Is used to show full menu of product in menu page and parse data from database.

Parameters:

Pic = the picture of product.

Name = the name of product.

Price = the price of product.

Desc = the description of product.

Return type void

5.1.2.5 Map(int x, int y)

Is used to show map and location of our company.

Parameters:

X = the x axis pointer that used in yandex.maps api.

Y = the y axis pointer that used in yandex.maps api.

Return type void

5.1.2.6 Cart_fixed(int id, int count, int total)

Is used to show just price in all pages except cart page.

Parameters:

Id = the id of product which have to be added to users cart.

Count = the count of product that have to be added to users cart.

Total = the total cost that user had before adding product to cart.

Return type void

5.1.2.7 Order(int id, int count)

Is used to make order.

Parameters:

Id = the id of product that has been oredered.

Count = the amount of product that has been ordered

Return type void

5.1.2.8 Show_order(int[] ids, int[] counts)

Is used to show all orders of our user.

Parameters:

Ids = the ids of all products that has been ordered, each has its respective order

Count = the count of all products that has been ordered, each has its respective order

(id[0] and count[0] is id of product and how many was ordered)

Return type void

5.2 PROCESS INTERFACE

5.2.1 Desktop Main process

Description: This process shows all graphical interface of the system

5.2.1.1 Process is created when the application started

5.2.1.2 Terminated when applications close button is pressed

5.2.1.3 All other threads will be killed if this main thread stops

5.2.2 Database Listener process

5.2.2.1 This thread is created after Main process acquires all information from database

5.2.2.2 Database listener process interacts with Panel Updater process

5.2.2.3 This process will be terminated automatically if Main thread of process is killed

5.2.3 Panel Updater process

5.2.3.1 This thread is created after Database Listener process

5.2.3.2 Mainly this thread interacts between Main thread and Database listener tread

5.2.3.3 Thread is terminated when one of other two threads terminates

6 Detailed Design

NOT REQUIRED <Java Docs to be used instead>

7 Design Rationale

7.1 DESIGN ISSUES

7.2 <ISSUE 1>

7.2.1 Description

7.2.2 Factors affecting Issue

7.2.3 Alternatives and their pros and cons

7.2.4 Resolution of Issue

7.3 <ISSUE 1>

7.3.1 Description

7.3.2 Factors affecting Issue

7.3.3 Alternatives and their pros and cons

7.3.4 Resolution of Issue

8 Traceability

No	Use Case/ Non-functional Description	Subsystem/Module/classes that handles it
1		
2		

FEEL FREE TO ADD APPENDICES AS NEEDED. UPDATE TOC BEFORE SUBMITTING