Система за електронна търговия Balkan Bay

Дизайн Модел

Версия1.0

История на промените

|  |  |  |  |
| --- | --- | --- | --- |
| **Дата** | **Версия** | **Описание** | **Автор** |
| 2014.03.20 | 1.0 | Първа версия на документа | Симеон Илиев |
| 2014.04.09 | 1.1 | Добавяне на диаграми | Симеон Илиев |

Съдържание

1. Представяне 4

1.1 Предназначение 4

1.2 Определения, акроними и съкращения 4

1.3 Връзки 4

1.4 Възможности 4

2. Подход 4

2.1 Дизайн шаблони 4

2.1.1 Model View Controller(MVC) 4

3. Дизайн 5

3.1 Дизайн на Изгледи 5

3.1.1 Технология преглед 5

3.2 Клас диаграми 7

3.2.1 Bbay entities 7

3.2.2 PayPal entities 7

3.2.3 Product Entities 8

3.2.4 User Relationship 9

Дизайн модел

# Представяне

## Предназначение

Предназначението на този документ е да представи дизайн решенията и това, как те ще бъдат имплементирани по време на разработка на системата - Система за електронна търговия Balkan Bay. Главната цел на документа е:

* Да даде ясна представа за това как ще бъдат разработени различните функционалности;
* Какви технологии, методи и принципи ще бъдат използвани при разработката.

Предвидената аудитория на документа е:

* Възложители;
* Разработчици;
* Тестери;
* Архитекти;
* Дизайнери на базата данни;
* Интерфейс дизайнери.

Документът ще бъде детайлизиран и развиван и по време на Разработката на проекта.

Решенията при изготвянето на дизайна на системата са взети въз основа на границите очертани от функционалните изисквания. В зависимост от тях са избрани подходящи технологии и методики за бързото и лесно изграждане на цялостната система.

Дефиниции, акроними и абревиатури Информация за използваните акроними, съкращения и техните определения можете да намерите в специализираният документ „Речник“

## Връзки

* „Софтуерна архитектура“;
* „Спецификация на софтуерните изисквания“.

## Възможности

# Подход

Документът е създаден във фаза „Детайлизиране“ и ще бъде развиван във фаза „Разработка“ от жизнения цикъл на проекта.

* През фаза „Детайлизиране“ в документа се описват подхода и решенията, които са взети относно архитектурата, базата данни, различните инструменти, които ще се използват за изграждане на системата;
* През фаза Разработка ще бъдат описани детайлно подходите, методите и различните решения при изграждането на отделните функционалности на системата. Ще се опишат алгоритмите за решение на възникнали проблеми по време на разработката, както и различните модели, използвани за изграждането на системата.

## Дизайн шаблони

### Model View Controller(MVC)

MVC е софтуерен модел за прилагане на потребителски интерфейси. Той разделя дадено софтуерно приложение в три взаимосвързани части, така че да се отделят вътрешни представяния на информация от начините, по които информацията е представена или се приема от потребителя. Controller компонента, на модела, се състои от данни на приложението , бизнес правила, логика, и функции. Model (модел) компонента може да бъде всеки изход, служещ за представяне на информация, като например диаграма. Множество гледки към една и съща информация са възможни, като бар-графика за управление и табличен изглед за счетоводители. В третата част от модела, администратора приема входна информация и я преобразува в команди за модела или изгледа.

Модела уведомява своите свързани View (изгледи) и контролери, когато е налице промяна в състоянието. Това уведомяване позволява изгледите да актуализират представянето си, а контролерите за промяна на наличния набор от команди.

Прозореца получава от контролера цялата информация, която е нужна за генериране на подходящото View за изход за потребителя.

Контролерът може да изпраща команди към модела, за да актуализира състоянието на модела (например, редактиране на документ). Тя също може да изпраща команди към съответния прозорец, за да се промени представянето на съответния прозорец .



Фигура 1 Model View Controller

# Дизайн

## Дизайн на Изгледи

### Технология преглед

Тъй като за разработка на системата се използват Java технологии, за създаването на различните потребителски интерфейси, беше избрана технологията Java Server Faces(JSF). Тя е Java технология за създаване на уеб-базирани приложения на основата на Java Servlet API и JSP. JSF е стандартизирана технология и е включена официално в Java Enterprise Edition 5. JSF включва:

* Компоненти - JSF е компонентно базирана технология. Компонентите служат за визуализиране на потребителския интерфейс. Всяка компонента запазва собственото си състояние, като състоянието може да се пази на сървъра или при клиента (клиентски браузър под вид на скрито поле, в което състоянието е сериализирано и записано чрез Base64 кодиране)
* Валидиране и конвертиране на входните данни от компонентите
* Стандартен набор от компоненти
* Поддръжка на интернационализиране
* Навигация между страниците
* Генерира на събития на сървъра в зависимост от действията от страна на клиента
* Възможност за създаване на потребителски библиотеки с JSP тагове
* Менажирани бийнове (Managed Beans) - представляват Java класове, които са дефинирани в конфигурационен файл и JSF се грижи те да бъдат създадени при извикването им, или връщането им ако вече са били създадени.
* Език за връзване на Java класове с компонентите (Expression Language)

#### Принцип на работа

Основната идея на технологията е да се имплементират компоненти, които създават потребителския интерфейс. Компонентите са преизползваеми, което означава, че веднъж написани те могат да се използват на различни места без да бъде повтарян или пренаписван техният код.

Компонентите представляват Java класове, които съдържат логика и генерират кода на потребителския интерфейс. Състоят се от три основни класа:

* ComponentTag – клас, който служи за обработка на атрибутите подадени към JSP тага на съответния компонент
* Component - в него се описва логиката на компонента
* Renderer - генерира изходният код (HTML в уеб приложенията)

Различните Renderer класове се описват в Render Kit, който може да бъде заменян по време на работа и това позволява един и същ код на приложението да генерира изходен код за различни видове устройства - уеб браузъри, PDA устройства, WAP за мобилни телефони.

#### Стандартни библиотеки

JSF предлага две стандартни компонентни библиотеки - JSF Core и JSF HTML.

##### JSF Core

* + Съдържа основни тагове за работа с JSF. Таговете от библиотеката JSF Core не са компоненти, а са помощни тагове за работа с компоненти.
  + f:view - оказва корена на дървото с компоненти. Трябва да бъде първата компонента в страницата.
  + f:subView – страници, които са вложени в текущата трябва да бъдат поставени в subView (има се в предвид влагане чрез jsp:include)
  + f:actionListener - добавя ActionListener към родителския компонент
  + f:attribute - добавя атрибут към родителския компонент
  + f:converter - служи за добавяне на конвертор към родителския компонент
  + f:convertDateTime - служи за конвертиране на дата записана в компонентът, в който конверторът е вложен
  + f:convertNumber - служи за конвертиране на числа записани в компонентът, в който конверторът е вложен
  + f:validator - служи за добавяне на валидатор към родителския компонент
  + f:validateLength - служи за добавяне на валидатор за дължина на въведеният текст в компонента, в който валидаторът е вложен
  + f:validateLongRange - служи за добавяне на валидатор за интервал на въведеното цяло число в компонента, в който валидаторът е вложен
  + f:validateDoubleRange - служи за добавяне на валидатор за интервал на въведеното дробно число в компонента, в който валидаторът е вложен
  + f:facet - начин на именуване на дъщерен компонент. Поставена във facet дъщерният компонент може да бъде извлечен от родителския, чрез името на facet.
  + f:loadBundle - служи за използване на ResourceBundle .properties файлове
  + f:param - служи за добавяне на параметър към заявката. Най-често се използва с h:commandLink
  + f:selectItem - служи за добавяне на елемент към компоненти за избор (компоненти, чиито имена започват с h:select. Пример: h:selectOneMenu)
  + f:selectItems - служи за добавяне на множество от елементи към компоненти за избор. Множеството от елементи се подава като Map или масив/списък с обекти от тип SelectItem.
  + f:verbatim - Главен недостатък на JSF е невъзможността в JSF компоненти да бъде вложен HTML код. За да стане това възможно HTML кодът трябва да бъде поставен във verbatim таг.

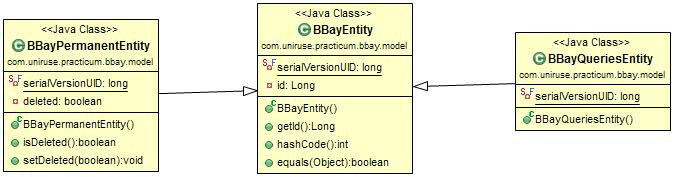
#### JSF HTML

#### Съдържа тагове за работа с основните JSF компоненти генериращи HTML код.

## Клас диаграми

### Bbay entities

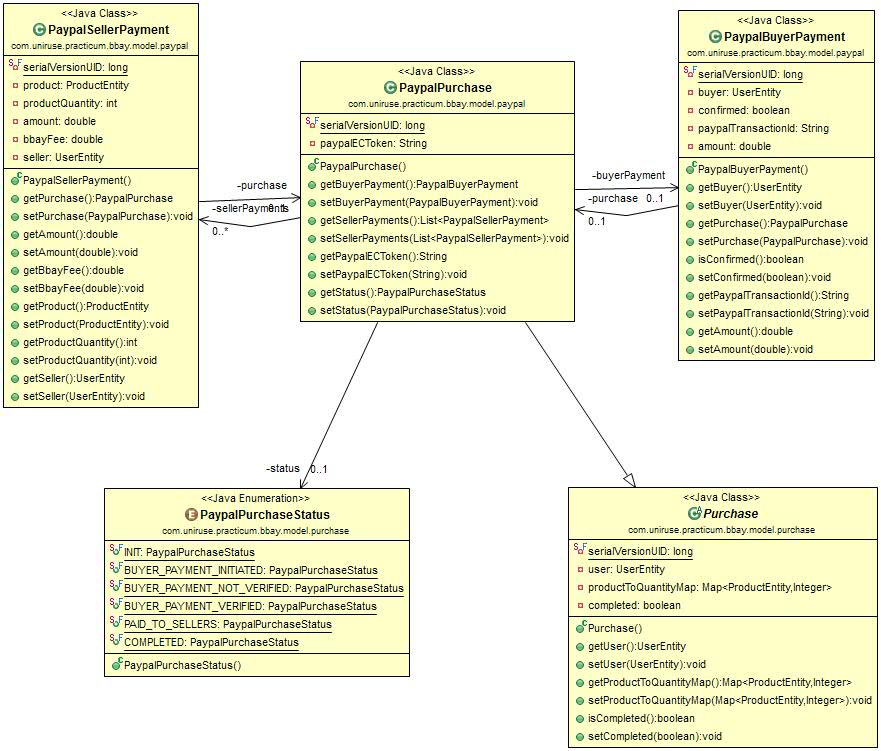
На Фигура 2 са показани трите основни класа за система Bbay. Главният за системата е BBayEntity.



Фигура 2

### PayPal entities

На Фигура 3 са показани класовете, които съдържат информация за PayPal, на различните потребители от една транзакцияи се съхранява информацията за транзакцията.



Фигура 3

### Product Entities

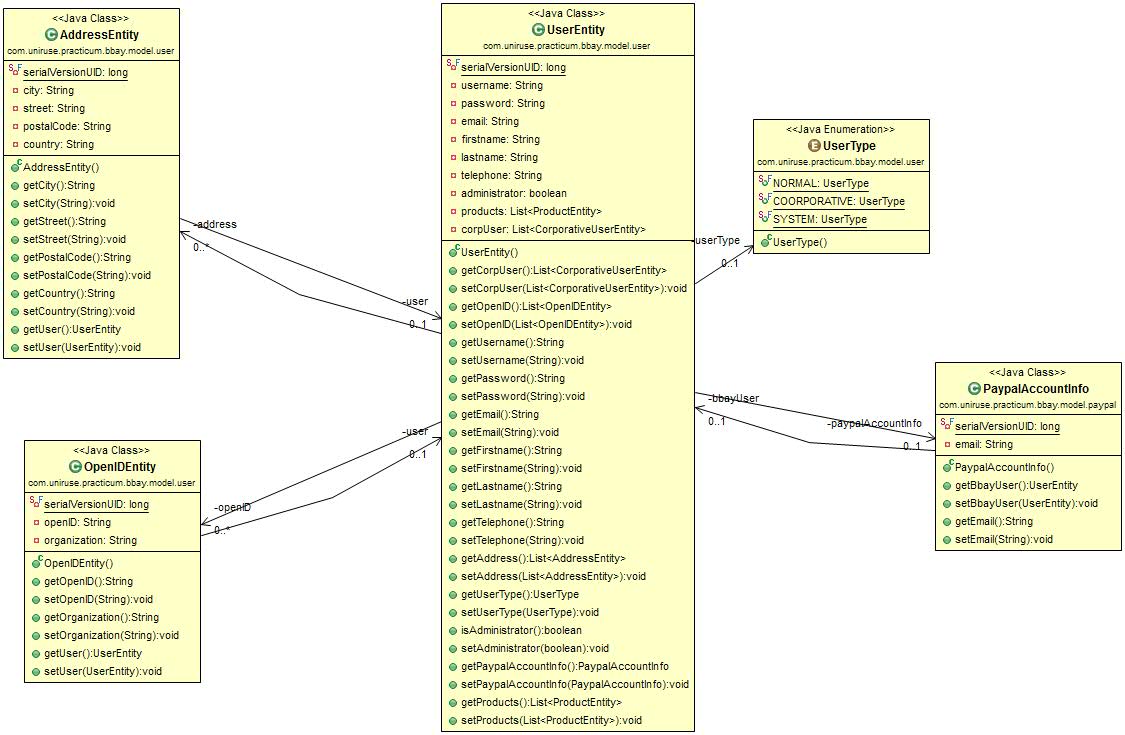
На Фигура 4 са показани класовете, съдържащи информация за продуктите.

### C:\newWork\bbay\bbay-api\src\main\java\com\uniruse\practicum\bbay\model\diagrams\ProductDiagram.jpg

Фигура 4

### User Relationship

На Фигура 5 е представена информацията, която се съхранява в системата за даден потребител и връзката между различните класове.



Фигура 5