Projektdokumentation

Kick-Off-Meeting am 28.02.12

Im Kick-Off-Meeting ist festgelegt worden, dass der Source Code über einen Subversion-Server geteilt wird. Außerdem wurde der Projektleiter ausgewählt. Weiterhin wurde die Arbeit auf die Mitarbeiter verteilt. Phillip Willms ist für die Utils zuständig und Thomas Trinklein ist für die Models zuständig. Die Klasse Reiseverwaltung wurde von Thomas Trinklein eingepflegt und in Kooperation mit Phillip Willms verbessert und verändert.

|  |  |
| --- | --- |
| Projektauftrag | |
| Projektname | BSFBS – Bus-Software für das Busunternehmen Sonnenschein |
| Projektleiter | Phillip Willms |
| Projektanlass | Das Busunternehmen wünscht eine Buchungssoftware, mit der Reisen und Kunden einfach verwaltet werden können |
| Projektziele | Siehe Lastenheft |
| Zu erarbeitende Ergebnisse | Funktionsfähige Software zur Buchung von Städtereisen gemäß den Anforderungen im Lastenheft |
| Randbedingungen | * Es werden vier Reisen angeboten * Diese Reisen finden in den Sommerferien statt * Die Busse fahren drei Wochen lang im Wochenrhythmus * Jeder Bus hat einen festgelegten Starttag |
| Termine und Meilensteine | 24. Februar: Projektstart und Kick-Off Meeting  02. März: Festlegen der Klassen  09. März: Testfälle aufstellen und durchlaufen  16. März: Die Software wird den Busunternehmen vorgestellt |
| Unterschriften | Auftraggeber Auftragnehmer  (Name, Unterschrift) (Name, Unterschrift) |

## Lastenheft

* Es wird eine Software für einen PC-Arbeitsplatz benötigt, über die die Belegung der vier Busse möglich ist. Eine Platzreservierung wird nicht gewünscht.
* Für jeden Bus sollen die Anzahl der zur Verfügung stehenden Plätze angezeigt werden.
* Eine neue Buchung soll automatisch mit einer neuen Buchungsnummer versehen werden.
* Eine Überbuchung soll nicht möglich sein. Dies soll durch eine Fehlermeldung angezeigt werden.
* Alle Teilnehmer einer Reise sollen in alphabetischer Reihenfolge aufgelistet werden können.
* Es soll eine Möglichkeit zur Stornierung einer Buchung bestehen. Auch dieser Vorgang soll automatisch einer fortlaufenden Nummer zugeordnet werden.
* Eine Unterdeckung soll verhindert und durch eine Fehlermeldung angezeigt werden.
* Mögliche Eingabefehler in den Buchungsdaten sollen korrigiert werden können.
* Alle Buchungen und Stornierungen sollen in einer log-Datei gespeichert werden.
* Die Daten sollen mit Excel und Word weiter verarbeitet werden und dazu im CSV-Dateiformat abgespeichert und geladen werden.

## Arbeitspakete

Modellierung von Klassen:

Thomas Trinklein

Dokumentation:

Thomas Trinklein

Philipp Willms

Testen:

Philipp Willms

Datenverwaltung:

Philipp Willms

Implementation der Nutzerinteraktion:

Philipp Willms

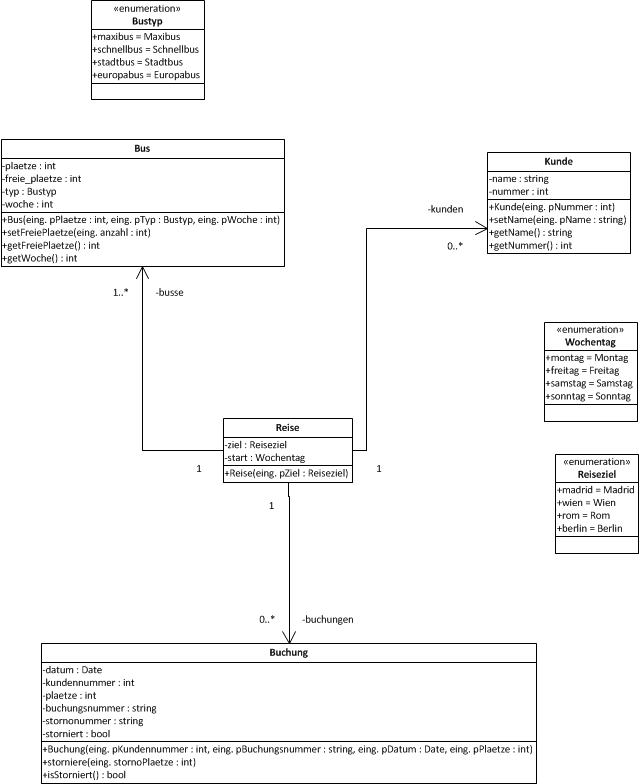
Thomas Trinklein

## Projektablaufplan

|  |  |  |  |
| --- | --- | --- | --- |
| Vorgangsnummer | Vorgangsbezeichung | Dauer (Tage) | Vorgänger |
| 1 | Erstellung eines Klassendiagramms | 1 | - |
| 2 | Implementation der Klassenfunktionalität | 7 | 1 |
| 3 | Implementation der Datenverwaltung | 8 | 1 |
| 4 | Javadoc | 2 | 2,3 |
| 5 | Testprotokoll erstellen und durchführen | 2 | 4 |
| 6 | Projektdokumentation | 2 | 5 |

## Klassendiagramm vor der Implementierung

Vor der Implementierung wurde dieses Klassendiagramm angelegt:



Das gesamte Programm wird später über die Reise gesteuert. Hierbei hat jede Reise null bis unendlich Buchungen, solange noch Platz im Bus ist. Außerdem hat jede Reise einen bis drei Busse, die an verschiedenen Starttagen losfahren, was in der Enumeration „Wochentag“ verwaltet wird. Weiterhin haben die Busse ein unterschiedliches Reiseziel, welches in der Enumeration „Reiseziel“ verwaltet wird. Außerdem gibt es verschiedene Bustypen, welche in der Enumeration „Bustyp“ aufgelistet sind.

Zu jeder Reise gibt es auch Kunden, jedoch gibt es maximal so viele Kunden, wie die Reise Plätze zur Verfügung stellt.

Der Konstruktor „+Bus(eing.pPlaetze : int, eing.pTyp: Bustyp, eing.pWoche : int);“ initialisiert die Klasse „Bus“.

Der Konstruktor „+Kunde(eing.pNummer: int);“ initialisiert die Klasse „Kunden“.

Der Konstruktor „+Reise(eing.pZiel: ReiseZiel);“ initialisiert die Klasse „Reise“.

Der Konstruktor „+Buchung(eing.pKundennummer: int, eing.pBuchungsnummer: String, eing.pDatumg: Date, eing.pPlaetze: int);“ initialisiert die Klasse „Buchung“.

Die Methode „+storniere(eing.pStornoPlaetze: int);“ storniert eine gewisse Anzahl an Plätzen.

Die Methode „+isStorniert(): bool“ findet heraus, ob die Buchung bereits storniert wurde.

Die anderen Methoden sind Getter- und Setter-Methoden, welche speziell geforderte Werte setzt und zurückliefert.