|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Projekttitel  **Programmierung eines Liefershops mit Datenbankanbindung** | | | | | | Datum  **01.12.2016** |
| GitHubprojektnummer  **06** | Name  **PizzaPizza** | Projektbeginn  **02.11.2016** | | | Projektende  **10.01.2017** | |
| Auftraggeber/in  **Prof. Dr. Oliver Braun** | | Projektleitung  **Yannic Fraebel, Nils Engelbrecht** | | | | |
| Statusbericht erstellt durch  **Yannic Fraebel, Nils Engelbrecht** | | Meilenstein / Projektphase  **1** | | | | |
|  | | **Projektstatus insgesamt** | | | | |
| **Ampel** |  | starke Probleme, besondere Aktivitäten erforderlich | | |
|  | Probleme vorhanden aber im Griff | | |
|  | kein Handlungsbedarf | | |

1. **Grad der Zielerreichung / Status des Prototyps**

Auf der „Startseite“ bietet die Applikation folgende Funktionen:

* Wahlmöglichkeit zwischen Kunde und Mitarbeiter (Rolle) anhand eines Dropdown-Menüs. Hierbei sind als Default-User ein Kunde „Emil“ und ein Admin namens „Padrone“ hinterlegt.
* Außerdem können sich neue Kunden eigenständig registrieren (dies wird in die Datenbanktabelle „Users“ übernommen).

Die „welcomeUser“ und „welcomeAdmin“-Seite beinhaltet folgende Funktionen:

* Die Kundenview bietet die Wahlmöglichkeit zwischen den vorhandene Kategorien Pizza, Getränke und Desserts.
* Einzelne Items können in Mengen und in einer variablen Größe (in cm bzw. l) bestellt werden. Der Preis wird mengen- und größenabhängig verbucht.
* Kunden können sich ihre bisherigen Bestellungen mit Vorgangsnummer, Datum und Details zur Bestellung ausgeben lassen.
* Mitarbeiter haben die Möglichkeit, sich alle registrierten Benutzer anzeigen zu lassen und alle bisherigen Bestellungen einzusehen.
* Kunden und Admins haben außerdem die Möglichkeit, sich abzumelden. (Redirect zur Startseite)
* Kategorisierung der User in Admins und Kunden erfolgt durch Setzen eines Admin Flags in der Users Tabelle.
* Preiskalkulation der Bestellung ist schon vollständig implementiert (sowohl intern als auch in JS um Benutzerfreundlichkeit zu gewährleisten)

1. **Aufgabenverteilung im Team**

|  |  |
| --- | --- |
| **Beschreibung** | **Bearbeiter** |
| Erstellen des Statusberichts | Yannic Fraebel |
| Grad der Zielerreichung | Yannic Fraebel |
| Aufgabenverteilung im Team | Yannic Fraebel/Nils Engelbrecht |
| Projektplan für die Restlaufzeit | Yannic Fraebel/Nils Engelbrecht |
| Risikoanalyse | Yannic Fraebel |
| Aufwandsnachweis | Yannic Fraebel/Nils Engelbrecht |
| Erstellen des Lastenhefts | Yannic Fraebel |
| Erstellung und Einbindung der Tabellen | Nils Engelbrecht |
| Kategorisierung der User durch ein flag | Nils Engelbrecht |
| Erstellung der Views | Nils Engelbrecht |
| Erstellung der Models | Nils Engelbrecht |
| Erstellung der Services | Nils Engelbrecht |
| Erstellung der Controller | Nils Engelbrecht |
| Implementieren der Preisberechnung | Nils Engelbrecht |
| Testen der Projektfunktionalität | Yannic Fraebel/Nils Engelbrecht |

1. **Plan für die Restlaufzeit des Projekts**

Geplant ist es die Restlaufzeit des Projekts maßgeblich grob dennoch präzise einzuschätzen. Nach dem SCRUM Verfahren werden nach und nach Funktionalitäten implementiert, integriert und sofort getestet, um einen weitere sinnvollen Verlauf der weiteren zu implementierenden funktionalen Anforderungen zu gewährleisten.

Folgende Implementierungsideen sind noch angedacht:

|  |
| --- |
| * Anzeige von Gesamtumsatz |
| * Allg. Auswertungen (🡪 Pizzen/ Bestellung) |
| * MA read/write Rechtekonzept |
| * Login über freies Textfeld und Passwortabfrage |
| * Bestellbestätigung per email falls vorhanden |
| * Weitere Auswertungsmöglichkeiten |
| * View Unterstützung für Bestellaufnahme von mehreren Produkten auf einmal sowie mehrere verschiedene Extras pro Item * (Evtl. Lösung durch Warenkorb Implementierung) |
| * Optimierung von User Registrierungsformular Error Handling z.B. check ob Name schon vorhanden |
| * Liefermöglichkeit je nach Kundenentfernung implementieren |
| * Lieferdauerberechnung und View Einbindung implementieren |

1. **Risikoanalyse**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| *Risikoart* | *Wahr-scheinlichkeit* | *Auswirkung* | *Ampel* | *Verantwortlicher* | *Beschreibung* | *Maßnahme(n)* |
| Datenverlust | 1 | 5 | 5 | Fraebel / | Verlust der Projektdaten | Regelmäßige Sicherung auf GitHub |
| Engelbrecht |
| Krankheit | 3 | 3 | 9 | Fraebel / | Teammitglied erkrankt | Telefonkonferenz zur weiteren Planung führen |
| Engelbrecht |
| Keine rechtzeitige Abgabe | 3 | 5 | 15 | Fraebel / | Projekt wird nicht rechtzeitig abgeben | Studienarbeit stärker priorisieren |
| Engelbrecht |
| Probleme Syntax Scala/HTTP/SQL | 2 | 4 | 8 | Fraebel / | Probleme beim Programmieren durch neue Sprachen | Tutorials lesen/schauen und Bücher im Eigenstudium |
| Engelbrecht |
| Probleme Dokumentation | 2 | 3 | 6 | Fraebel | Schwierigkeiten beim Erstellen des Lastenhefts und des Projektplans | Orientierung an Lastenheften, die von Unternehmen online gestellt wurden. |
| Anforderungen | 1 | 2 | 4 | Engelbrecht | Falsche oder überflüssige Programmteile werden implementiert | Funktionale Anforderungen lesen und Absprache mit Herr Prof. Dr. Oliver Braun |
| Mehrfachbearbeitung | 2 | 4 | 8 | Fraebel / | Arbeitsaufträge werden parallel bearbeitet und gepusht | Verwenden der VCS Möglichkeiten |
| Engelbrecht |
| Pragmatische Programmfehler | 3 | 2 | 6 | Fraebel / | Applikation arbeitet zu umständlich, das MVC-Modell wird nicht eingehalten | Gegenseitiges Überprüfen um Fehler zu vermeiden |
| Engelbrecht |

1. **Aufwandsnachweis**

|  |  |  |
| --- | --- | --- |
| **Datum** | **Nils Engelbrecht** | **Yannic Fraebel** |
| 16.11.2016 | - Clonen des zur Verfügung gestellten Git-Repo’s  - Erste Einarbeitung in das Play-Framework  - Einbindung eines FavIcons | - Clonen des zur Verfügung gestellten GIT- Repo’s  - Erzeugen der Projektdateien für IntelliJ.  - Erste Einarbeitung in das Play-Framework |
| 17.11.2016 | - Erstellung einiger Forms  - Unterscheidung MA vs. Kunde | - Verstehen des bereits gegebenen Projektcodes |
| 18.11.2016 | - Registrierungs Validierung  - Datenbank getUser implementiert | - Recherche zum Thema Statusbericht |
| 19.11.2016 | - Kategorien und Items eingeführt - Bestellfunktionalität  - Default Datenbanktabellen Values festgelegt  - dynamisches Kunden-Menü angelegt | - Anlegen des Statusberichtes |
| 20.11.2016 | - Bestellanzeige je KundenID | - Recherche zum Thema Lastenheft  - Anlegen des Projektplans |
| 25.11.2016 | - Bestellanzeige optimiert  - Re-lokation der Klassen/ Methoden je nach Funktionalität (Separation of Concerns)  - SQL Default Skripte nach Tabellen getrennt | - Testen des Prototyps |
| 26.11.2016 | - Bestellvorgang Bug gelöst | - Anlegen des Lastenhefts |
| 27.11.2016 | - Cent in Euro Anzeige erstellt  - Preiskalkulation implementiert | - Fertigstellen und Korrekturlesen des Statusberichtes und des Lastenheftes |
| 28.11.2016 | - Versuch Extras Form dynamisch via JS einzufügen  - Optionale Extra fields implementiert |  |
| 29.11.2016 | - vollständige backend Extra Bestellaufgabe realisiert, trotz ViewForm Problemen |  |
| 30.11.2016 | - Fertigstellung Aufwandsnachweis |  |
| 01.12.2016 | - Überarbeitung von Lastenheft & Statusbericht  - Anpassung der View für die  Extra-Auswahl |  |
| 18.12.2016 | - Einführung von Sessions  - Passwortfunktionalität hinzugefügt  - Login- und Registrierungsform Validierung eingeführt (bereits vorhanden/ existiert nicht etc.) |  |
| 22.12.2016 | - Userverwaltung Funktionalität implementiert (Name und Admin Änderung) |  |
| 23.12.2016 | - Userverwaltung Unterscheidung zwischen Kunde und MA  - Form Error Message Display erneuert |  |