



# **Ergebnis- dokumentation**

## **PLANINATOR 3000**

für die Schwietzer GmbH

LAHMMP AG

---

# ERGEBNISDOKUMENTATION

## PLANINATOR 3000

28.11.2021

### Inhalt

Kurze Zusammenfassung .....	2
1. Einführung .....	3
2. Projektstatus .....	4
3. Projekt-Dokumentation.....	5
4. Technische Dokumentation.....	8
4.1 Struktur des Systems.....	8
4.2 Verwendete Technologien und APIs .....	8
4.3 Funktionsweise des Planninator3000.....	9
4.4 Get started with the Planinator3000 .....	12
5. Review (Dozentenansicht) .....	14

## KURZE ZUSAMMENFASSUNG

### Einführung

Einleitung zur Ergebnisdokumentation

### Projektstatus

Aufsummierung aller Projektkosten zu neuem Gesamtwert von 27.494,85 €

### Projekt-Dokumentation

Beschriebener Projektablauf und Unterscheidung von ursprünglicher Projektplanung. Grundsätzlich wurde Projekt erfolgreich abgeschlossen. Dabei lief das Vorgehen mit Verantwortlichkeiten und Meilensteine gut. Trotzdem gab es einige Herausforderungen bei dem Programmieren, die erfolgreich gelöst wurden.

### Technische Dokumentation

Grundaufbau und technische Einzelheiten werden anhand von einzelnen Beispielen erläutert. Außerdem wird eine Anleitung zur Einrichtung benötigter Ressourcen des Planinators für Linux und Windows geliefert.

### Review (Dozentenansicht)

Projektdokumentation, verwendete Tools und Zeitplanung können noch optimiert werden. Gut liefen die Verantwortlichkeiten, Kommunikation und Umgang mit Herausforderungen.

## 1. Einführung

Sehr geehrte Damen und Herren,

Vielen Dank, dass Sie sich für unseren Planinator 3000, ihr Budgetverwaltungssystem, entschieden haben. Wir hoffen, dass unser gemeinsames Projekt in einem zufriedenstellenden Rahmen absolviert wurde. Eine detaillierte Anweisung zur Nutzung des Programms finden Sie in unserem angehängten Benutzerhandbuch.

Unser junges innovatives Team hat sich viel Mühe gegeben ein herausragendes Produkt mit Herzblut zu liefern, ganz nach dem Motto „eine strahlende Zukunft mit unseren Services“.

Bei abschließenden Fragen wenden Sie sich bitte an ihren Ansprechpartner Adam Weinschenk unter der Telefonnummer +49 711 99796866 oder per Mail an [support@lahmmp.de](mailto:support@lahmmp.de).

Viel Erfolg auf ihrem zukünftigen Weg  
Ihr LAHMMP Projektteam

## 2. Projektstatus

In dem Zwischenbericht von dem 23.10.2021 wurde eine Erhöhung des Budgets für den Prototyp I dargelegt. In diesem Bericht wurde durch einen Forecast gezeigt, wie die Projektkosten sich bei gleichbleibenden prozentualen Mehraufwänden entwickeln würden. Gleichzeitig wurde umrissen, warum dies vermutlich nicht der Fall sein wird. Die finalen Projektkosten bewegen sich innerhalb des Forecastrahmens. Es sind noch einige Herausforderungen auf uns zugekommen, die überwiegend durch den eingeplanten Puffer aufgefangen wurden. Allerdings sind dennoch einige zusätzliche Arbeitsstunden hinzugekommen.

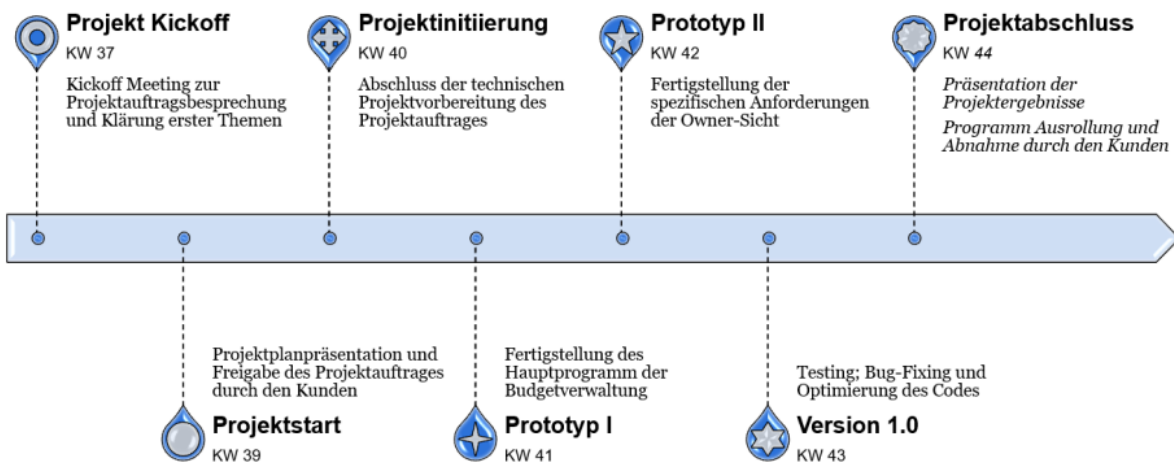
Projektkostenübersicht					
Meilenstein	Stunden			Differenz in €	Kosten in €
	SOLL	IST	Differenz		
Projektstart	108	108	0	0	6.852,60
Projektinitiierung	39	39	0	0	2.474,55
Prototyp I	92	146	+54	+ 3.426,30	5.837,40 + 3.426,30 = 9.263,70
Prototyp II	38	50	+12	+761,40	2.411,10 + 761,40 = 3172,50
Version 1.0	27	35	+8	+507,60	2.220,75 + 507,6 = 2.728,35
Projektabschluss	55	55	0	0	3.489,75
<b>Finale Projektkosten</b>	<b>239</b>	<b>293</b>	<b>+54</b>	<b>+ 3.426,30 + 1.290</b>	<b>22.778,55 + 3.426,30 + 1.290 = 27.494,85</b>
Ursprünglich kalkulierter Budgetrahmen					22.778,55

### 3. Projekt-Dokumentation

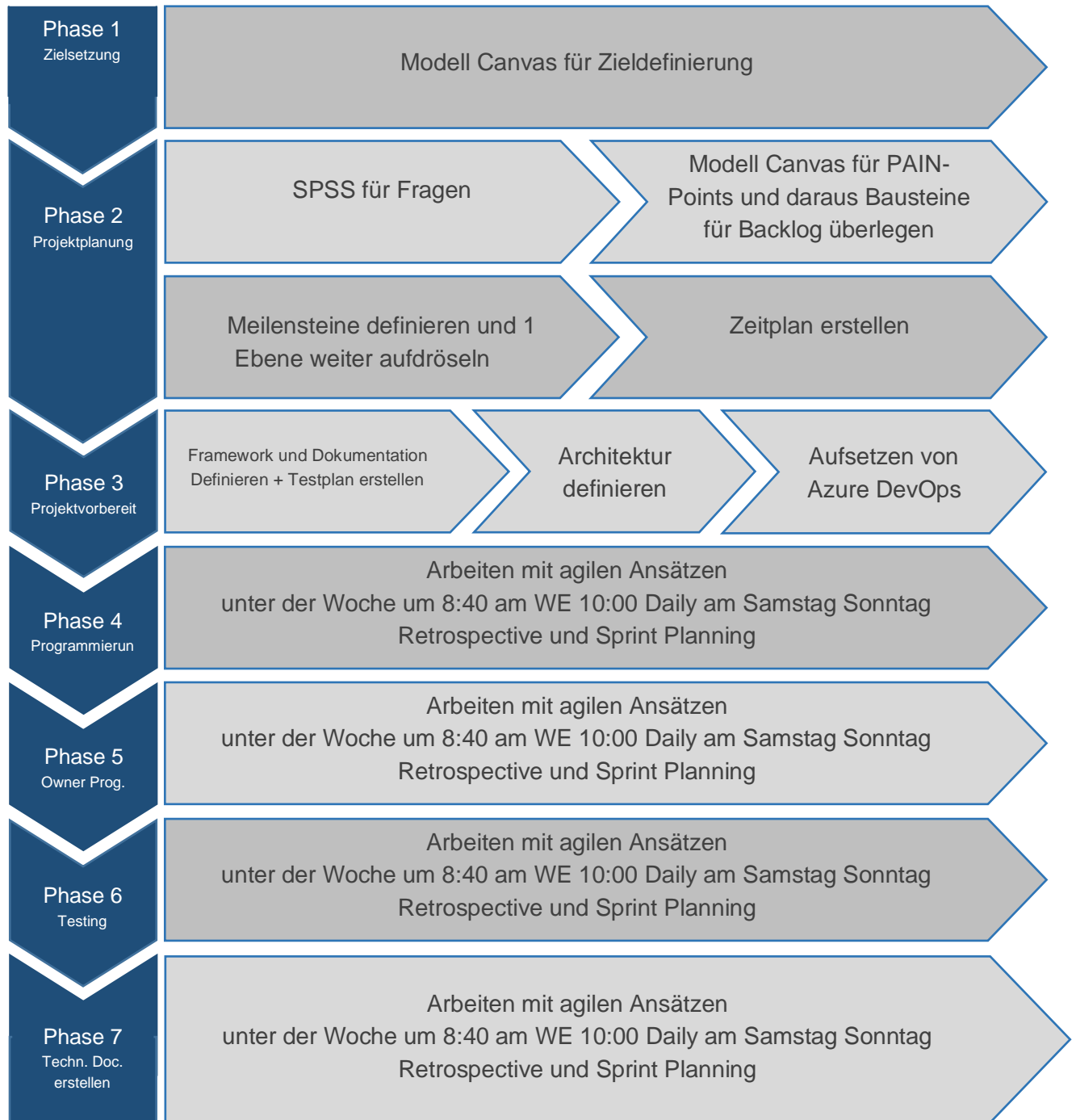
Um den Projektablauf komplett verstehen zu können werden hier noch einmal die Grundgedanken des Projektes aufgezeigt.

Zunächst einmal ist keine klassische Rollenaufteilung gegeben gewesen, sondern den Mitgliedern des Projektes wurden verschiedene Bereiche als Verantwortlichkeit übergeben. Dies bedeutete eine höhere Flexibilität in den verschiedenen Phasen des Projektes, sodass immer die gegebenen Ressourcen voll umfänglich eingesetzt werden können und eine schnellstmögliche Beendigung des Projektes gegeben war.

Das Projekt wurde, wie gerade erwähnt, in verschiedene Phasen eingeteilt, die einen jeweiligen Meilenstein besitzen, sodass der Schwietzer GmbH auf Wunsch war ein Feedback gegeben werden konnte, ob der Zeitplan eingehalten werden kann und ein Zwischeninkrement an Sie ausgegeben werden konnte. Der Plan sah wie folgt aus:



Innerhalb der verschiedenen Phasen sollte auf unterschiedliche Art und Weisen gearbeitet werden, je nachdem was am effizientesten war, um ein qualitativ hochwertiges Produkt herzustellen. Kernstück des Projektes sind die Phasen, in denen programmiert wurde. Hierbei wurden agile Ansätze verwendet, um den Kunden in den Prozess mit einbeziehen zu können. Dies war nur einmal in Form eines Zwischenberichts gegeben.



Am Anfang des Projektes wurde so geplant, dass bis zwei Wochen vor der Kundenpräsentation, das Programm zum Großteil fertiggestellt wird. Es wurde, wie in Kapitel 3 schon genannt, ein Meilensteinplan erstellt, in welchem alle wichtigen Ereignisse aufgeführt wurden. Das Projekt wurde dadurch in sieben verschiedene Phasen unterteilt.

Die Phasen 1-3 verliefen weitgehend positiv, das Team schaffte es eine klare Zielsetzung festzusetzen und mit der Projektplanung anzufangen. Alles wurde genau durchgeplant, zum Beispiel, wann welche Sprints anstehen.

Die Projektvorbereitung lief größtenteils im ganzen Team ab. Es wurde stets jeder Mitarbeiter zu einem festgelegten Termin gerufen, um das weitere Vorgehen der im Projekt notwendigen Schritte zu besprechen und darüber abzustimmen. Hierbei gibt es zu erwähnen, dass durch die verteilten Verantwortlichkeiten, sich jeder auf seine eigenen Bereiche spezialisierte und bis zum nächsten Termin detailliert Bescheid wusste, was ansteht und wann was zu erledigen ist. Dazu hatte diese Person auch die Möglichkeit verschiedene Arbeitspakete an die verfügbaren Projektbeteiligten zu verteilen. Durch die aufgeteilten Verantwortlichkeiten schaffte es das LAHMMP Team auch, sauber durch die Projektdurchführung zu kommen.

Bis zum ersten weiteren Kontakt mit der Schwietzer GmbH in KW41, verlief alles wie geplant. Kurz vor Fertigstellung des Zwischenberichts, bei dem Prototyp 1 bereits fertig programmiert sein sollte, wurde wie vorher schon befürchtet, bemerkt das wir ein wenig zurückhängen mit dem Zeitplan. Daraufhin wurde ein neuer Zeitplan erstellt und zusätzlich angefallene Aufwände dem Kunden mitgeteilt. Es ist dabei zu erwähnen, dass die Manager der Schwietzer GmbH jedes Mal erreichbar waren und uns auch gut von ihrer Seite aus unterstützt haben. Das Zusammenspiel zwischen der Schwietzer GmbH und LAHMMP war aus Sicht von LAHMMP sehr gut.

Nach Erstellung des neuen Zeitplans, welcher auch im Zwischenbericht an die Schwietzer GmbH übergeben wurde, fingen wir mit der Programmierung von Prototyp 1 und 2 an. Bis zur Kundenpräsentation stand dann die erste Version des Planinators.

Nach Fertigstellung der ersten Version des Planinators setzt sich das Programmiererteam ans Bugfixing.

Zusammenfassend lässt sich sagen das die am Anfang gewählten Tools uns dabei geholfen haben das Projekt gut und schnell fertigzustellen. Dazu ist es so, dass das ganze Team Hand in Hand zusammengearbeitet hat. Es gab nahezu keine Auseinandersetzungen und falls jemand falls jemand Hilfe benötigte, bekam er diese.

Bezüglich der Arbeitsweise im Projekt lässt sich sagen, dass zuerst versucht wurde agil zu arbeiten, es aber schlussendlich ein Wasserfallvorgehen mit Agilen Ansätzen war.

Dazu wurde sich am Anfang des Projektes jeden morgen zu einem Daily getroffen, bei dem aktuelle Themen besprochen wurden. Neben den Dailys gab es auch jeden Sonntag um 10 Uhr ein Weekly bei dem, ein recap der Woche gemacht wurde und am Ende die nächste Woche durchgeplant wurde. Eine beispielhafte Dokumentationen der Dailys und Weeklys sind im Anhang zu finden, falls Bedarf besteht, diese zu besichtigen. Diese wurden jedoch nur am Anfang dokumentiert und später nur noch mündlich besprochen.

Final lässt sich sagen das der Projektverlauf größtenteils sehr positiv verlaufen ist. Es gab zwischendrin ein paar zeitliche Diskrepanzen, welche jedoch durch Absprachen im Team wieder aufgeholt werden konnten.



## 4. Technische Dokumentation

Der Planinator3000, wurde mit den modernsten Technologien entwickelt. Ziel war es den Planinator so zu entwickeln, dass Ressourcen wie CPU, RAM und ähnliches sparsam und effizient verwendet werden. Ebenso war der Hintergedanke, dass mit einer einfachen und überschaubaren Logik gearbeitet wird, damit auch später, dank dieser, ohne Probleme neue Features eingebunden werden können.

### 4.1 STRUKTUR DES SYSTEMS

Um das System bestmöglich übersichtlich und verständlich zu strukturieren, wurde sehr viel Wert auf die Package Struktur gelegt. Getrennt wurde nach der technischen Logik und nach Systemspezifizierung. In der Abb. 1 ist die Package Struktur zusehen in der erkennbar ist, dass je nach Bereich und Logik ein Package angelegt wurde.

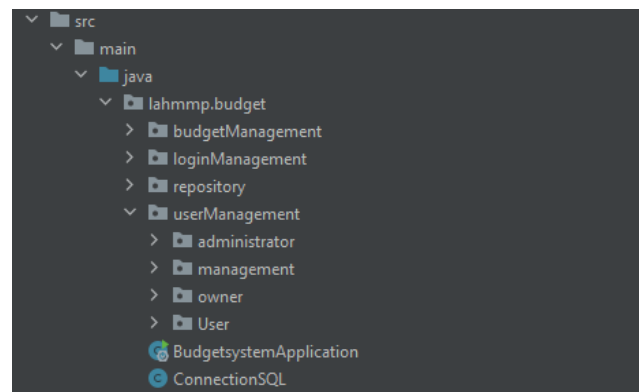


Abbildung 1: Package Struktur

Hinweis: Sie erhalten das fertige Programm „Planinator3000“ Budgetmanagement\_System\_Main (ohne die in der Entwicklung eingesetzten Testmethoden und keine alten Entwicklungsversionen).

### 4.2 VERWENDETE TECHNOLOGIEN UND APIS

Für die Grundsprache wurde Java gewählt, daher beruht die gesamte Business Logik dieser Sprache. Java arbeitet plattformenunabhängig und kann daher auf jedem Rechner mit der passenden Laufzeitumgebung genutzt werden.

Eines der wichtigsten Frameworks für das Programm ist Spring. Dieses ist ein modular aufgebautes Framework für Java. Es gilt als sehr leichtgewichtig und ist ein Open Source-Projekt. Ziel des Frameworks ist, die Komplexität der Java-Plattform deutlich zu reduzieren. Dazu bietet Spring bereits grundlegende Technologien fürs Web, z. B. WebSockets, REST Web Services, somit genau das, was wir brauchen.

Ein weiteres Framework, was genutzt wurde ist Hibernate. Bei dieser handelt es sich um ein Framework zur Abbildung von Objekten auf relationalen Datenbanken für die Programmiersprache Java. Es wird auch als Object Relational Mapping Tool bezeichnet. Hibernate ist in der Programmiersprache Java implementiert und steht als Open Source zur freien Verfügung.

Damit Hibernate die Daten aus der Datenbank bekommt, wird eine Schnittstelle benötigt. Dazu wird JPA eingesetzt. JPA ist ein API für Datenbankzugriffe und objektrelationales Mapping. Exakt formuliert wurde Spring Data JPA eingesetzt. Spring Data JPA, Teil der größeren Spring Data-Familie, macht es einfach, JPA-basierte Repositories zu implementieren. Es erleichtert die Erstellung von Spring-basierten Anwendungen, die Datenzugriffstechnologien verwenden.

Damit das Spring-Java Backend Daten an das Frontend schickt und dieses mit den Daten arbeiten kann, wird eine Template-Engine benötigt. Template-Engines haben hierfür den passenden Ansatz, mit der Aussicht, ohne umständliche Kompilierung in Servlets und ohne Managed Beans auszukommen. Für den Planinator3000 wurde sich für Thymeleaf entschieden. Es kann mit dem Funktionsumfang zum Beispiel von Java Server Pages (JSP) aufnehmen und dabei performant, erweiterbar und einfach integrierbar sein.

Hinweis: Thymeleaf zeigt teils Fehler im Code des Programms an, was aber keine Fehler sind. Der Code funktioniert ordnungsgemäß. Dies Problem konnten wir nicht beheben.

```
<!-- Ende Tabelle-->
<!-- Fehlermeldungen für den Benutzer basierend auf logischen Restriktionen -->
<div>
  <p th:if="${session.err.isEmployeeNumberErr()}" class="fehler">Fehler bei der Mitarbeiternummer</p>
  <p th:if="${session.err.isNoOwnerFoundErr()}" class="fehler">Kein Owner gefunden</p>
  <p th:if="${session.err.isPlanntamountErr()}" class="fehler">Fehler bei dem geplanten Budget</p>
  <p th:if="${session.err.isArchivatedErr()}" class="fehler">Fehler bei dem Archivierungsstatus</p>
  <p th:if="${session.err.isDateErr()}" class="fehler">Fehler bei dem Ablaufdatum</p>
</div>
```

Abbildung 2: Thymeleaf zeigt teils Fehler teils nicht

### 4.3 FUNKTIONSWEISE DES PLANNINATOR3000

Das System wird durch die Klasse „BudgetsystemApplication“ gestartet, da dort die Main mit der Annotation „@SpringBootApplication“ gestartet.

```
@SpringBootApplication
public class BudgetsystemApplication {

    public static void main(String[] args) {
        SpringApplication.run(BudgetsystemApplication.class, args);
    }
}
```

Abbildung 3: SpringBootApplication

Wenn das System gestartet wird, gibt die Konsole folgendes aus:

```
2021-11-11 19:35:35.956 INFO 22292 --- [main] lahmp.budget.BudgetsystemApplication : Starting BudgetsystemApplication using Java 17.0.1 on DESKTOP-334HP81 with PID 22292
2021-11-11 19:35:35.958 INFO 22292 --- [main] lahmp.budget.BudgetsystemApplication : No active profile set, falling back to default profiles: default
2021-11-11 19:35:36.364 INFO 22292 --- [main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in DEFAULT mode.
2021-11-11 19:35:36.408 INFO 22292 --- [main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 39 ms. Found 4 JPA repository interfaces.
2021-11-11 19:35:36.727 INFO 22292 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (http)
2021-11-11 19:35:36.732 INFO 22292 --- [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2021-11-11 19:35:36.732 INFO 22292 --- [main] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.53]
2021-11-11 19:35:36.818 INFO 22292 --- [main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2021-11-11 19:35:36.819 INFO 22292 --- [main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 831 ms
2021-11-11 19:35:36.919 INFO 22292 --- [main] o.hibernate.jpa.internal.util.LogHelper : HH0000204: Processing PersistenceUnitInfo [name: default]
2021-11-11 19:35:36.948 INFO 22292 --- [main] org.hibernate.Version : HH0000412: Hibernate ORM core version 5.4.32.Final
2021-11-11 19:35:37.035 INFO 22292 --- [main] o.hibernate.annotations.common.Version : HCAN0000001: Hibernate Commons Annotations {5.1.2.Final}
```

Abbildung 4: Konsolen Ausgabe (Systemstart)

Die Konsole sagt uns, dass das System gestartet ist. Es gibt uns Informationen wie und das vier Repositories, genauer gesagt vier Spring Data JPA Repositories, geladen werden. Wir haben je ein Repository für das Budget, den Forecast, die History und für die User. Um nun auf das System zugreifen zu können, startet Spring automatisch einen Apache Tomcat Server. In der Standardeinstellung startet Spring den Server auf den Port 8080, jedoch kann dies nach Wunsch ohne Probleme angepasst werden. Benötigen wir soeben Daten z. B. für das Anzeigen aller Budgets, wird durch zwei verschiedene Methoden die Daten ein- und ausgelesen. Einmal geschieht es durch den JDBC Driver, die Verbindung wird in der Klasse „ConnectionSQL“ (siehe Abb. 1) aufgebaut. Eine andere Methodik, die verwendet wurde, ist über Spring und JPA. In der Klasse „ConnectionSQL“ wird eine Verbindung durch die Methode „connectToDatabase“ hergestellt. Darin werden die 3 wichtigen Parameter url, user und password übergeben. Diese Parameter dienen dazu, damit das Programm weiß, aus welchen Schema/Tabelle es Daten beziehen bzw. senden soll. In unserm Fall ist die url=“jdbc:mysql://localhost/db\_planinator“, user=“root“, password=“Sonne“. Mit denselben Parametern wird auch über Spring und JPA verbunden. Die Parameter werden jedoch in der application.properties eingetragen mit:

- spring.datasource.url=jdbc:mysql://localhost:3306/db\_planinator
- spring.datasource.username=root
- spring.datasource.password=Sonne

Der Grundaufbau, wie die Logik ausgeführt wird, wurde einheitlich gestaltet, mit Spring. In der Abb. 5 ist zu erkennen, dass jede Rolle und der User selbst, drei verschiedene Klassen hat. Alle haben einen Controller, einen Service und eine rollenspezifische Klasse (Owner, Administrator, Management und User). Die spezifischen Klassen dienen dazu z. B. für die Objekterstellung und um Attribute zu definieren. Im Service wird die Businesslogik der jeweiligen Rollen programmiert. Dort werden z. B. die Daten aus der Datenbank geholt und festgelegt wie mit denen umgegangen wird. Im Controller wird das Frontend gesteuert.

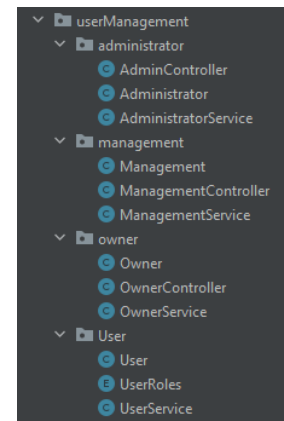


Abbildung 5: Klassenaufbau und Arten

```
@GetMapping("/Budget_Uebersicht_Management")
public String Budget_Uebersicht_Management(Model model, HttpSession session) {
    if (testAccess.testAccess(session, UserRoles.MANAGEMENT)) {
        Budget budgetid = new Budget();
        model.addAttribute( attributeName: "budgetid", budgetid);
        model.addAttribute( attributeName: "listBudget", managementService.getAllBudgets());
        return "management1";
    }
    return "redirect:/";
}
```

Abbildung 6: Methode für Budget Ausgabe

In der Abb. 6 ist eine Controller-Methode dargestellt. Der Grundaufbau ist wie folgt: Es wird eine Annotation vergeben je nach Methode. Wenn die Methodik oder die Businesslogik Daten an die Webseite senden soll, dann wird die Annotation @GetMapping verwendet. Sollte die Methodik bzw. Businesslogik Daten bekommen, z. B. ein neuer User oder ein Budget angelegt wird, dann wird eine @PostMapping Annotation benötigt. Beide Annotationen sollen HTTP Anfragen abfangen. Der Name der Anfrage wird als String Parameter übergeben. Die Methode selbst ist wie eine gewöhnliche Java-Methode aufgebaut. Was jedoch wichtig ist, ist der mit übergebene Parameter „Model“. In diesem Model-Objekt werden alle Daten „eingepackt“, die von der Service-Klasse übergeben werden, um sie der Webseite zu weiterzugeben. Mit dem return, wird der Methode gesagt, wohin oder welche Webseite (HTML Dokument) es öffnen soll, mit dem Model. In dieser Methode wird z. B. erst überprüft, ob der User eine Managementrolle besitzt. Hat der User die Berechtigung, diese Aktion durchzuführen, wird die Seite „management1“ aufgerufen. Sollte der User keine Berechtigung haben, dann wird der User auf dieselbe Seite wieder verwiesen durch „redirect:/“.

## 4.4 GET STARTED WITH THE PLANINATOR3000

**Empfohlenes Betriebssystem: Linux**

**Benötigte Datenbank: MySQL (MySQL-Server) neuste Version**

**Optionale Tools: MySQL Workbench**

Bevor Sie das System starten, ist es wichtig, dass der MySQL Server läuft. Um das System bestmöglich testen zu können, empfehlen wir Ihnen, die mit zur Verfügung gestellten MySQL Daten zu nutzen. Bitte führen Sie erst die Datei DB\_Schwietzer aus und danach die Datei Testdaten\_Schwietzer. Sind nun alle Daten abgespeichert, ist es wichtig, dass Sie die richtigen Parameter im Code selbst eingetragen werden. Wir empfehlen Ihnen, über die MySQL Workbench eine neue Connection zu erstellen und sich an den bereits codierten Daten zu orientieren. In diesem Fall wäre es:

- url=jdbc:mysql://localhost:3306/db\_planinator
- username=root
- password=Sonne

Sollte dies nicht möglich sein für Sie, dann können Sie auch die Parameter nach ihrer Verbindung umcodieren. Es ist wichtig, dass Sie an beiden Stellen im Code dies tun. Einmal in der application.properties File und ebenfalls in der ConnectionSQL Klasse.

Läuft der MySQL Server mit den Beispieldaten und wurde je nach dem, die Datenbank Parameter richtig angepasst, dann können wir nun das System starten. Nach dem Sie die Zipdatei heruntergeladen habe, extrahieren Sie bitte diese an ihrem Wunschort.

### Für Linux:

Öffnen Sie nun ein Terminal im Ordner „Budgetmanagement\_System\_Main“ oder navigieren sie sich dahin. Mit dem Befehl: „mvn spring-boot:run“ starten sie das Programm.

### Für Windows:

Installieren Sie sich bitte eine aktuelle Version der IntelliJ IDE. Hier öffnen Sie nun das Projekt über „File“→“Open“.

Nachdem Sie das Projekt eingebunden haben, laden Sie bitte das Maven Script, siehe Abb. 7.

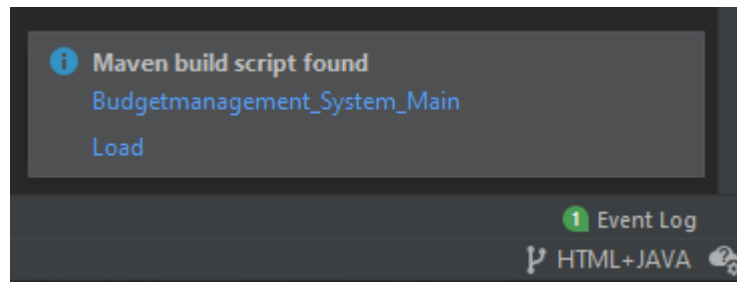


Abbildung 7: Maven Build laden

Anschließend öffnen Sie das Projekt links in der Übersicht und öffnen dort die Java Datei BudgetsystemApplication.java(src/main/java/lahmmp/budget/BudgetsystemApplication.java). Hier müssen Sie als nächstes bestätigen, dass das Maven Project ordnungsgemäß ist und sie diesem vertrauen.

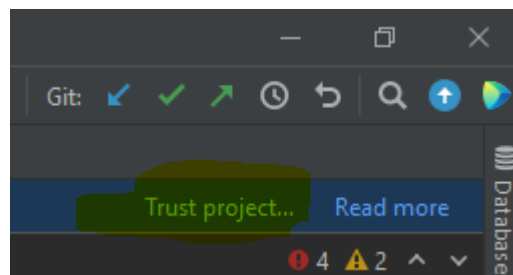


Abbildung 8: Maven Project vertrauen

Installieren Sie zuletzt, falls benötigt etwaige notwendige Plugins, wie etwa das JDK. Haben Sie alles vorbereitet klicken Sie bitte auf den Playbutton, siehe Bild rot umrandet.

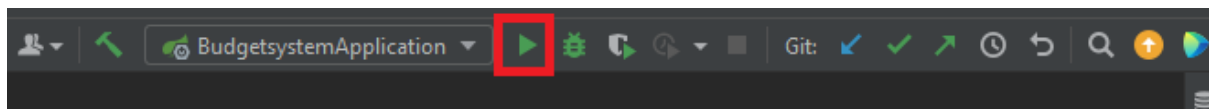


Abbildung 9: Playbutton

### Webseite öffnen:

Das System läuft von Haus aus auf den Port 8080. Sollten bereits andere Dienste auf diesen Port laufen haben, ist es möglich den Port zu ändern. Um Ihren Wunsch Port auszuwählen, tragen Sie in der application.properties ein: „server.port=[Port]“. Öffnen Sie nun Ihren Wunschbrowser und öffnen sie den Localhost mit ihren ausgewählten Port.

### Support:

Sollte das Programm nicht starten, melden Sie sich bitte bei dem Product Owner Adam Weinschenk.

## 5. Review (Dozentenansicht)

In diesem Projekt hatten wir sehr viele verschiedene Punkte welche gut liefen. Zum einen war es die Kommunikation im Team. Diese hat über Teams sowie, über WhatsApp sehr gut funktioniert. Jeder stand fast immer zur Verfügung, wenn er braucht wurde. Dazu muss gesagt werden, dass die Einteilung der Verantwortlichkeiten, hier auch sehr viel dazu beigetragen hat. Dadurch, dass wir feste Verantwortlichkeiten für jeden Punkt hatten, konnten wir direkt die bestimmte Person ansprechen und somit die Kommunikation auch auf einem Minimum halten.

Durch die verschiedenen Verantwortlichkeiten haben wir es auch geschafft die Aufgaben schnell und schlau zu verteilen. Wir haben uns zur Aufgabenverteilung oft im Team zusammengetroffen und dann das Ganze am Anfang in Confluence oder Azure DevOps eingetragen. Somit wusste jeder immer was er machen musste und welcher Schritt gerade ansteht, oder was schon erledigt wurde.

Dazu muss gesagt werden, dass das Ganze nicht ohne unsere sehr gute Projektplanung am Anfang funktioniert hätte. Wir haben uns am Anfang sehr lange daran gesetzt einen Projektplan zu erstellen. Durch den Projektplan gelang es uns eine sehr gute Übersicht zu schaffen, wann was erledigt werden musste.

Bezüglich des Backends müssen wir sagen, dass der Datenbankaufbau sehr gut verlief. Wir hatten wenig Schwierigkeiten die Datenbanken aufzusetzen und zu füllen. Neben den Datenbanken lief auch das Frontdesign sehr gut. Wir haben hierzu einen grundlegenden Aufbau erstellt und dann das Ganze nach Absprache im Team umgesetzt. Hierzu müssen wir auch noch erwähnen, dass die erste Anbindung zwischen Front- und Backend auch sehr gut funktioniert hat, was uns sehr erleichtert hatte, da wir am Anfang nicht wussten, ob die Verbindung überhaupt wie gewünscht funktioniert.

Wir sehen jedoch noch einiges Potential in der Verwendung von Confluence und Azure DevOps. Wir haben Confluence fast gar nicht mehr zur Planung benutzt, nur noch, um die Aufgabenverteilung der Form halber nachträglich noch schriftlich festzuhalten. Nebenbei ist auch noch zu erwähnen, dass wenn wir unsere Programmiersprachen (Spring und Thymeleaf) besser gekannt hätten, wir viel schneller ein besseres Programm erstellt hätten. Auch trotz unseres guten Projektplans, kam es zu einer schlechten Zeitplanung. Wir mussten zwischendurch in unser Projekt bemerken, dass wir zeitlich nicht unseren aktuellen Plan einhalten konnten. Deswegen musste somit eine neue Zeitplanung erstellt werden, welche wir dann im Nachhinein gut einhalten konnten. Der letzte Punkt, welcher nicht so gut verlief, war die Projektdokumentation. Diese ist bei uns auch zu wenig passiert, da wir wie gesagt fast kein Confluence oder Azure DevOps benutzt haben.

Zusammenfassend haben wir auch vieles im Projekt gelernt, zum einen haben wir neue Programmiersprachen und neue Programmiertechniken kennengelernt. Hierzu zählen auch erweiterte Restriktionen im Code.


Zudem hatten wir eine persönliche Weiterbildung in unserem Team, und zwar ist dieser Person aufgefallen, dass die Aufgabenverteilung sehr gut funktioniert hat. Es muss nicht alles kontrollieren, sondern das Team kann sich auch auf die anderen verlassen.

Dazu haben wir in der Vorlesung gelernt, wie eine gute Softwarepräsentation erstellt und präsentiert wird. Nebenbei haben wir auch noch gelernt, wie eine richtige Zeitplanung im Team aussieht und diese am besten in Bezug auf die verschiedenen Teamfähigkeiten setzt.





# ANHANG

## Beispielhafte Dokumentation eines Weekly:



Weeklys



Verantwortlichkeitsbereich:		Maximilian Schumm		
Besprochene Meilensteine:		Prototyp I		
Anwesende Personen:		 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	 <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/>
Maximilian Schumm	<input checked="" type="checkbox"/>			
Adam Weinschenk	<input checked="" type="checkbox"/>			
Lars Nordmann	<input checked="" type="checkbox"/>			
Harpreet Singh	<input checked="" type="checkbox"/>			
Paul Sevecke	<input checked="" type="checkbox"/>			
Meike Loh	<input checked="" type="checkbox"/>			
Datum:	16.10.2021	Uhrzeit:	11.45 Uhr	
Ort:	online	Protokollant:	Meike Loh	

### Kurzzusammenfassung des Meetings:

#### Anhänge

### Fertig gestellte Punkte seit dem letzten Meeting:

- Die meisten Kernfunktionen im Backend
- Datenbank aufsetzen
- Frontend-Schema

### Offene Diskussionspunkte:


1. Vorgehen der nächsten Woche
2. Form des Zwischenberichts

### Getroffene Entscheidungen:


1. Aufteilung der Aufgabenbereiche in Zweier-Teams
  - a. Gegenseitiger Austausch mit dem Partner und selbstständige Organisation untereinander
  - b. Offene Kommunikation bei Zeitproblemen etc.
  - c. Fokus auf *Zwischenbericht-Team* und *Fertigstellung Prototyp I – Team*; bei Bedarf springt *Prototyp II Backend – Team* ein, um bei Fokusaufgaben zu unterstützen




2. Zwischenbericht als (3-20 seitige) PDF mit Fließtextanteil (Vorgaben von Kunden):  
Ziel 4-5 Seiten in dem gleichen Format wie unser Projektplan

## Beispielhafte Dokumentation eines Retro:



Retro



Verantwortlichkeitsbereich:		Maximilian Schumm		
Besprochene Meilensteine:		Erster Prototyp		
Anwesende Personen:		 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input checked="" type="checkbox"/>
Maximilian Schumm	<input checked="" type="checkbox"/>			
Adam Weinschenk	<input checked="" type="checkbox"/>			
Lars Nordmann	<input checked="" type="checkbox"/>			
Harpreet Singh	<input checked="" type="checkbox"/>			
Paul Sevecke	<input checked="" type="checkbox"/>			
Meike Loh	<input checked="" type="checkbox"/>			
Datum:	16.10.2021	Uhrzeit:	11.05 Uhr	
Ort:	online	Protokollant:	Meike Loh	

**Kurzzusammenfassung des Retros**

Festgehaltene Probleme:

- Probleme mit der Entwicklungsumgebung IntelliJ (wegen unerwarteten Verhalten)
- Probleme mit Git (ein paar Merge Konflikte)
- Tickets wurden nicht von jedem konsequent verschoben

Umgang mit den Problemen:

- Gegenseitig Hilfe beanspruchen, wenn nötig
- Git im Terminal benutzen und stärkere Trennung der Branches in denen programmiert wird
- Im Team darauf hinweisen und für die Zukunft festhalten



Retro



	Bereich	Arbeitspakete	Bewertung
Maximilian Schumm	Scrum-Master	Grundlage Admin coden Vorbereitung Retro und Sprint	
Adam Weinschenk	Product-Owner	Grundlage Management coden	
Lars Nordmann	Frontend	Grob-Schema	
Harpreet Singh	Backend	Login aufsetzen	
Paul Sevecke	Datenbank	DB aufsetzen Beispieldaten eintragen	
Meike Loh	Testing	Grundlage Owner Coden	

### Vorläufige Agendapunkte für das nächste Sprint-Planning

- Frontend fertig stellen und Funktionen anbinden
- Aufstellung des Zwischenberichts

## Beispielhafte Dokumentation in Azure DevOps:

Fallstudie Budgetplanung Team

Board Analytics View as Backlog

Issues

To Do Doing 1/5 Bug 0/5 Ready for Review 0/5 Done

New item

79 Owner Spezifikation  
Harpreet Singh  
State To Do  
0/6

61 Kern des Budgetsystems programmieren  
Harpreet Singh  
State To Do  
5/5

49 Datenbank designen und aufsetzen  
Paul Sevecke  
State To Do  
2/2

48 Aufbau der Datenbank diskutieren  
Paul Sevecke  
State To Do  
2/2

86 Frontend & Backend Verbindung  
Lars  
State Doing  
0/1

47 Transfer auf Programmbereiche  
Lars  
State Done  
0/1

71 Zwischenbericht  
Maximilian Schumm  
State Done  
0/7

45 Idee für generelle Konzeption entwickeln und zeichnerisch darstellen  
Lars  
State Done  
1/5

46 Präsentieren und verbessern  
Lars  
State Done  
2/2

LAHMMP Delivery Plan

Teams

Fallstudie Budg...  
Epics

Sprint 1 6.10. - 16.10.  
Sprint 2 17.10. - 23.10.  
Sprint 3 24.10. - 30.10.

4 Prototyp I  
To Do  
Issue 3/5 60%

6 Version 1.0  
To Do

5 Prototyp II  
To Do  
Issue 0/1 0%

7 Projektabschluss  
To Do

Fallstudie Budg...  
Issues

Sprint 1 6.10. - 16.10.  
Sprint 2 17.10. - 23.10.  
Sprint 3 24.10. - 30.10.