

Intranet

Studienarbeit

Abteilung Informatik
Hochschule für Technik Rapperswil

Herbstsemester 2016

Autoren: Claudia Saxer, Frank Jordi
Betreuer: Jürg Jucker
Projektpartner: Intigena, Mettmenstetten

Abstract

Ausgangslage

Die Intigena Produktions AG plant in ihrem Industrie 4.0 Projekt die Automatisierung ihrer Produktionsstrasse für die Windelproduktion. Dieses Projekt umfasst verschiedene Aspekte, unter anderem die Labo-
rautomatisierung.

Die Qualitätssicherung wird in einem an die Produktionsstätte angegliederten Labor gemacht. So ist der Laufweg zwischen Kontrolleuren und den Maschinen respektive Produkten kurz und es kann in nützlicher Zeit reagiert werden, falls Qualitätsprobleme auftreten. Dennoch benötigen die durchgeführten Tests Zeit und der Informationsfluss ist nicht digitalisiert.

Vorgehensweise

In einer ersten Phase soll deshalb das Erfassen der Testresultate digitalisiert werden. Aktuell werden alle Angaben auf Papier erfasst und nachträglich in ein Excel-Arbeitsblatt eingefügt. Nun soll die Eingabe direkt aus dem Labor heraus in die Datenbank erfolgen. Die erfassten Daten sollen auch gleich im Labor visualisiert werden, um die Entwicklung der Resultate einfach verfolgen zu können. So sollen die Mitarbeiter im Labor frühzeitig negative Entwicklungen der Qualität erkennen und entsprechend reagieren können.

Ergebnis

Umgesetzt wurde das Projekt mit asp.net MVC welches aus einer Intranet-Rahmenapplikation sowie dem Module „Labor / QS“ besteht. Und einer geeigneten Datenbankstruktur auf dem SQL-Server.

Die Rahmenapplikation erlaubt das aktivieren und deaktivieren von verschiedenen Modulen durch den Administrator, auch wird die Navigation dynamisch pro User generiert um nur die, für ihn berechtigten Module darzustellen. Die Anmeldung und Berechtigungseinschränkungen laufen über das Active Directory von Windows. Die Kommunikation zwischen Datenbank und Applikation läuft via Entity Framework.

Das Labor beinhaltet die Eingabe von manuellen Messwerten sowie die Auswertungen. Nur über die Eingabe der Fertigungsauftrag Nummer, mit welcher Windeln und Binden zu einem Auftrag gebunden werden, bekommt der Tester die Testübersicht für den Artikel für die aktuelle Schicht. Alle Werte welche vorher mühsam eingegeben werden mussten und schon über den Auftrag und die Zeit berechnet werden können, werden generiert. Je nach Berechtigung kann der Tester nun Tests erstellen, editieren oder löschen. Werte welche nicht im erwarteten Range liegen werden hervorgehoben. Zu Qualitätsproblemen kann der Tester simple Meldungen hinterlassen. Damit die Entwicklung der Daten einfach nach zu verfolgen sind, besteht ein Dashboard sowie eine Historie.

Inhalt

1	Management Summary	6
1.1	Ausgangslage	6
1.2	Vorgehen und Technologien	6
1.3	Ergebnisse	6
1.4	Ausblick.....	6
2	Einleitung.....	7
2.1	Problemstellung, Vision	7
2.2	Ziele	7
2.3	Rahmenbedingungen	7
2.4	Methode / Vorgehen.....	7
3	Software Architektur	8
3.1	MVC 5.....	8
3.2	Inversion of Control	10
3.2.1	Übersicht	10
3.2.2	IoC Container mit Autofac.....	11
3.2.3	MEF Evaluation	12
3.3	Source Code	13
3.3.1	Solution / Projects.....	13
3.4	Schichtenmodell.....	16
3.4.1	Version 1 (Startidee).....	16
3.4.2	Version 2	16
3.4.3	Version 3	17
3.5	Build	17
3.5.1	Appvoyer	17
3.5.2	Cake.....	17
3.5.3	Slack	18
3.6	Web Technology	18
3.6.1	Gulp	18
3.7	Routing.....	18
3.8	Security	20
3.8.1	Authentifikation.....	20
3.8.2	Autorisation	20
3.9	Implementation (Entwicklung und Test).....	21
3.9.1	Automatische Testverfahren	21
3.9.2	Manuelle Testverfahren	24
4	Shell	25
4.1	Einleitung	25
4.1.1	Admin Tools	25
4.2	Datenbank.....	26
4.2.1	Datenbank Version 1	26
4.2.2	Datenbank Version 2	29
4.2.3	Generic Repository.....	30

4.3	Model	32
4.4	Package Diagramm	32
4.5	Navigation	33
4.5.1	View	33
4.5.2	Controller	33
4.5.3	Service	33
4.5.4	Bll	33
4.5.5	Diagramm	33
4.6	Logging	34
4.7	Projekt Diagramm	37
4.8	User Design	37
4.8.1	Mockups und Endergebnis	38
5	Labor	40
5.1	Einleitung	40
5.2	Datenbank	40
5.2.1	Labor	40
5.2.2	Intigena	43
5.2.3	Instron	47
5.3	Modell	48
5.4	Package Diagramm	48
5.5	Projektdiagramm	49
5.6	Sequenzdiagramm	50
5.6.1	Aufruf eines Testformulars zum Editieren	50
5.6.2	Speichern eines Testformulars	50
5.7	User Design	52
5.7.1	Startseite	52
5.7.2	Dashboard	52
5.7.3	Testerfassung	53
5.8	User Ablaufdiagramm	56
6	Testplan	57
6.1	Shell	57
6.2	Labor	60
6.2.1	Zugriffsrechte	60
6.2.2	Testerfassung	62
6.2.3	Dashboard	78
6.2.4	History	81
7	Zusammenfassung	82
7.1	Ergebnisse	82
7.2	Ausblick	82
7.2.1	Erweiterungsmöglichkeiten Shell	82
7.2.2	Erweiterungsmöglichkeiten Labor-Modul	82
7.3	Verbesserungsvorschläge	83
7.3.1	Server Side Caching	83
7.3.2	Navigation	83

8	Verzeichnisse	84
8.1	Glossar und Abkürzungsverzeichnis.....	84
8.2	Abbildungen	84
8.3	Tabellen	87
8.4	Literatur und Quellenverzeichnis	87
9	Danksagung.....	88
10	Anhang.....	89
10.1	Projekt Management	89
10.1.1	Code Auswertungen.....	89
10.2	Testbericht	90
10.2.1	Unit Test.....	90
10.2.2	Usability Test.....	90
10.2.3	System Test	90
10.3	Projektdiagramm	94
10.4	Package Diagramme	95
10.4.1	Package Diagramm Shell	95
10.4.2	Package Diagramm Labor	96
10.5	Klassen Diagramme	97
10.5.1	Intranet.X (Shell)	97
10.5.2	Intranet.Common.....	98
10.5.3	Intranet.Definition	104
10.5.4	Intranet.Bll	104
10.5.5	Intranet.Dal.....	106
10.5.6	Intranet.Model	107
10.5.7	Intranet.ViewModel.....	107
10.5.8	Intranet.Web.....	109
10.5.9	Intranet.Labor.Model	111
10.5.10	Intranet.Labor.Bll	115
10.5.11	Intranet.Labor.Dal.....	122
10.5.12	Intranet.Labor.Definition	123
10.5.13	Intranet.Labor.ViewModel.....	126
10.6	Separate Dokumente	128

1 Management Summary

1.1 Ausgangslage

Diese Studienarbeit befasst sich mit der Planung und Umsetzung einer Intranet-Applikation für die Firma Intigena.

Für die Mitarbeiter soll ein Firma internes Intranet zur Verfügung stehen, in dem sie mit spezifischen Modulen arbeiten können.

Die Authentifikation soll über ein bereits vorhandenes Active Directory geschehen.

Ein Modul für das Labor soll bereits mitgeliefert werden, in denen die Labormitarbeiter ihre Testresultate eingeben können und die Leitung sich einen Überblick über die Testergebnisse beschaffen kann. Optional sollten die Testresultate, welche automatisch von der Instron Testmaschine im Labor erstellt werden, geparkt und auch im Modul angezeigt werden.

1.2 Vorgehen und Technologien

Die Durchführung wurde in zwei Teilen aufgeteilt, in eine Shell Entwicklung und eine Labor Modul Entwicklung. Die beiden Teile wurden je in eine Analyse und Architekturphase und einer Entwicklungsphase unterteilt.

Am Ende die beiden Teile wurde noch Zeit eingeplant für die Dokumentation, das Testen, und für allfällige Bug fixes.

In der Analyse und Architektur Phase wurden jeweils die Anforderungen mit dem Auftragsgeber und dem Betreuer erarbeitet sowie die Architektur definiert.

Basierend auf der Analyse und Architektur wurden dann in der Entwicklungsphase jeweils die verschiedenen Komponenten entwickelt. Entwickelt wurde das Intranet mit ASP.NET MVC 5 und für die Datenbank kam ein SQL Server 2014 zum Einsatz.

1.3 Ergebnisse

Bis auf die Testresultate der Instron Maschine wurden alles in der Ausgangslage Beschriebene in dieser Studienarbeit entwickelt und getestet. Die Integration der Instron-Testresultate hätte den Rahmen dieser Arbeit gesprengt.

1.4 Ausblick

Da die Intranet-Applikation durch Module erweitert werden kann, könnte es noch viele Nachfolgeprojekte geben, in denen weitere Module entwickelt werden. Auch das Labor Modul könnte noch erweitert werden, da die Anforderungen nur Inkonsistenz- und Babywindeln beinhalteten, aber im Labor auch noch Tampons getestet werden.

2 Einleitung

2.1 Problemstellung, Vision

Diese Studienarbeit wurde durch die Firma Intigena, ein Anbieter von Hygieneartikeln, und Jürg Jucker in Auftrag gegeben.

Im Labor von Intigena, in dem Mitarbeiter die produzierten Waren testen, werden die Testresultate in nicht ganz einfach verständliche Excel-Arbeitsblätter eingetragen. Diese Excel-Arbeitsblätter werden dann an den Leiter des Labors gesendet, welcher nachschaut, ob noch alles in Ordnung ist. Da aber es aber immer mehrere einzelne Dateien sind, braucht dies viel Zeit und macht es dem Leiter nicht einfach einen Überblick über die Produktion zu gewinnen.

2.2 Ziele

Das Ziel dieser Arbeit ist es, ein durch Module erweiterbares Intranet und ein dazugehöriges Labor Modul zu entwickeln.

Das Labor Modul soll das in der Problemstellung angesprochene Thema vereinfachen. Der Leiter des Labors soll nicht mehr alle Excel-Dateien öffnen müssen, sondern nur noch ein klick im Intranet betätigen um sich einen Überblick über die Testresultate zu verschaffen. Die Labormitarbeiter sollen nicht mehr die unübersichtlichen Excel-Arbeitsblätter ausfüllen, sondern die Testresultate direkt im Intranet eingeben. Zusätzlich sollte man im Falle einer Kundenreklamation einfacher zu den entsprechenden durchgeführten Produkttestresultaten gelangen und nicht mehr wie bisher sich durch die Excel Dateien durcharbeiten.

2.3 Rahmenbedingungen

Das Intranet muss die neuste Version des Internet Explorers sowie des Mozilla Firefox unterstützen. Zusätzlich muss das Intranet auf Deutsch sein sowie einfach und verständlich für Benutzer, welche nicht viel Computererfahrung besitzen.

2.4 Methode / Vorgehen

In diesem Dokument wird auf die Software Architektur und die eingesetzten Technologien für die Entwicklung eingegangen. Im weiteren Verlauf folgt noch die Dokumentation für die Shell und für das Labor Modul sowie der Testplan für den Systemtest.

3 Software Architektur

3.1 MVC 5

Die Applikation ist mit ASP.net MVC 5 aufgebaut.

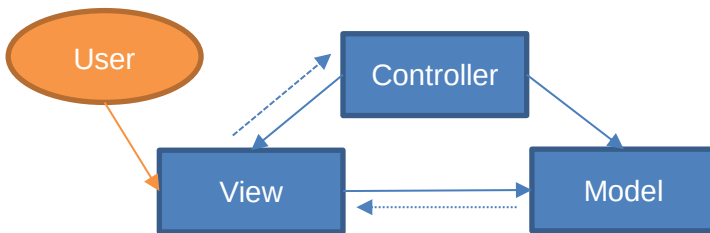


Abbildung 1: MVC simple

Die Abbildung zeigt ein simples MVC Pattern Beispiel.

Die View ist das Userinterface, welches Daten vom Model darstellt.
Das Model ist der Data Access Layer, welcher Datenbank Operationen ausführt.
Der Controller kontrolliert die Beziehung zwischen der View und dem Model, es erhält den Input vom User, kommuniziert mit dem Model und entscheidet was anzuzeigen ist.

Zusätzlich benutzen wir das MVVM Pattern.



Abbildung 2: MVVM

Das ViewModel stellt die Präsentation Logik dar.
Das Model stellt die Geschäftslogik und die Daten dar.
Die View stellt die Präsentation Ansicht dar, welche bei uns cshtml Files sind.

Das Model informiert das ViewModel über Datenänderung.
Das ViewModel gibt die aufbereiteten Daten weiter an das Model.
Die View enthält data binding auf das ViewModel.

Kombiniert sieht das nun bei uns etwa wie folgt aus.

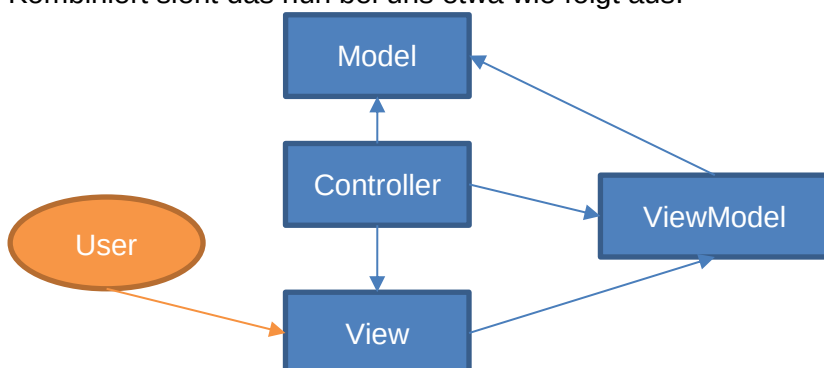


Abbildung 3: MVC + MVVM

Der Unterschied zum MVC ist, dass wir eine zusätzliche Abstraktion mit dem ViewModel haben. Die View stellt nun nicht mehr die Daten vom Model, sondern vom ViewModel Daten dar.

Für eine schöne Abstrahierung haben wir unsere Applikation noch weiter abstrahiert mit den gängigen Schichten.

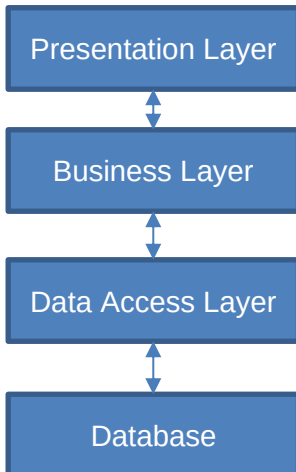


Abbildung 4: Schichtenarchitektur

Der Presentation Layer ist für die Präsentation verantwortlich.
Der Business Layer ist für die Business Logik verantwortlich.
Der Data Access Layer ist für den Datenbankzugriff verantwortlich.

Alle Patterns zusammen ergeben unsere Projekt Architektur, welche wir für eine für eine gut strukturierte Asp.Net MVC 5 Applikation einsetzen.

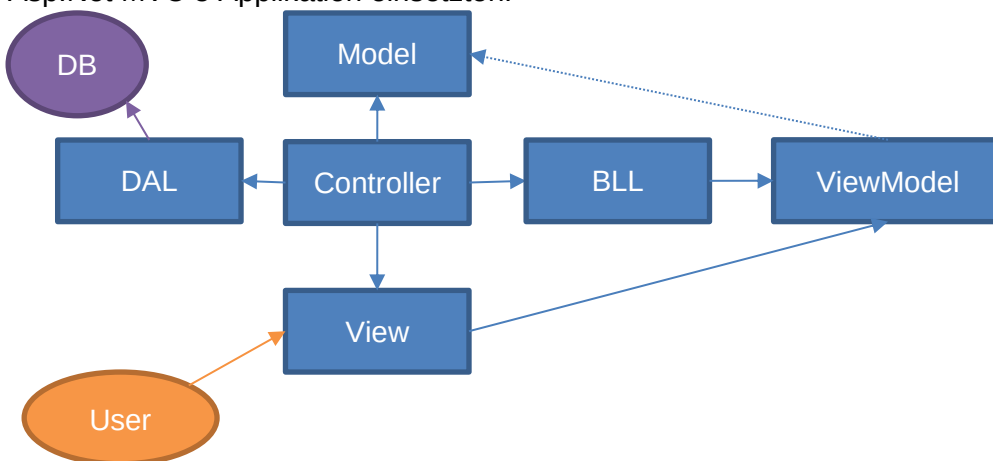


Abbildung 5: Final MVC Architektur

3.2 Inversion of Control

3.2.1 Übersicht

Beim Thema Inversion of Control gibt es DIP, IoC, DI und IoC container welche man unterscheiden muss.¹

Dependency Inversion Principle (DIP)

DIP ist ein Grundsatz, welcher in der Software Architektur benutzt wird.

Der Grundsatz besagt, dass höhere Module keine Anhängigkeiten zu low level Module besitzen sollen. Beide Module haben Abhängigkeiten via Abstraktion.

Inversion of Control (IOC)

IOC ist ein Pattern welches benutzt wird um den Ablauf, Dependencies und Interfaces zu invertieren.

Dependency Injection (DI)

DI ist die Implementation von IoC, um Dependencies zu invertieren.

IoC Contrainer

Der IoC Container ist das Framework welches DI macht.

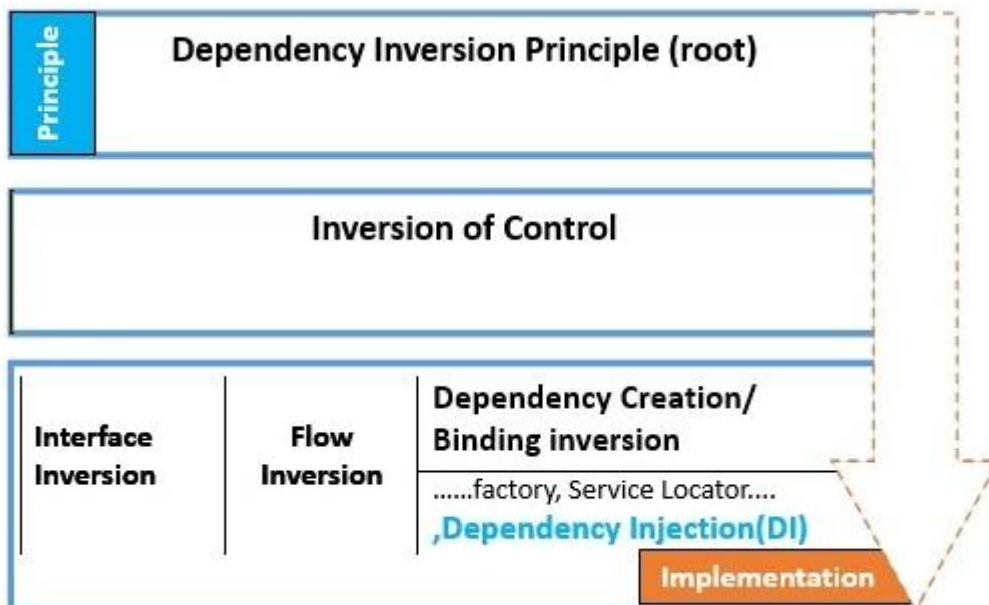


Abbildung 6: DIP, IoC und DI ²

Die Abbildung zeigt auf, wie alles zusammen funktioniert. DI ist jedoch nicht die einzige Art von Dependency Creation, welche wir hier benutzen. DIP zuoberst ist das Prinzip für Software Design. IOC ist ein Weg wie man DIP umsetzen kann und gibt Methoden vor, wie man es umsetzen kann. Wenn man Dependency Creation umsetzen möchte, kann man dies mit Dependency Injection Implementieren.

¹ Siehe [9]

² Siehe [7]

3.2.2 IoC Container mit Autofac

Wir benutzen Autofac als Inversion of Control Container. Autofac ist somit unsere Implementation für Dependency Injection.

In Autofac wird wie folgt ein Container erstellt.

```
var builder = new ContainerBuilder();
```

In diesem Container werden die verschiedenen Komponenten registriert.

```
builder.RegisterType<HomeBll>()
    .As<IHomeBll>()
    .PropertiesAutowired()
    .InstancePerRequest();
```

Wenn wir die Funktionalität von HomeBll brauchen, deklarieren wir das Interface welches wir im Definition Projekt abgelegt haben. Somit haben wir keine direkte Abhängigkeit auf die Implementation sondern nur auf das Interface, da uns die genaue Implementation nicht interessiert, sondern nur die Funktionalität.

```
/// <summary>
// Gets or sets the bll for the home.
/// </summary>
public IHomeBll HomeBll { get; set; }
```

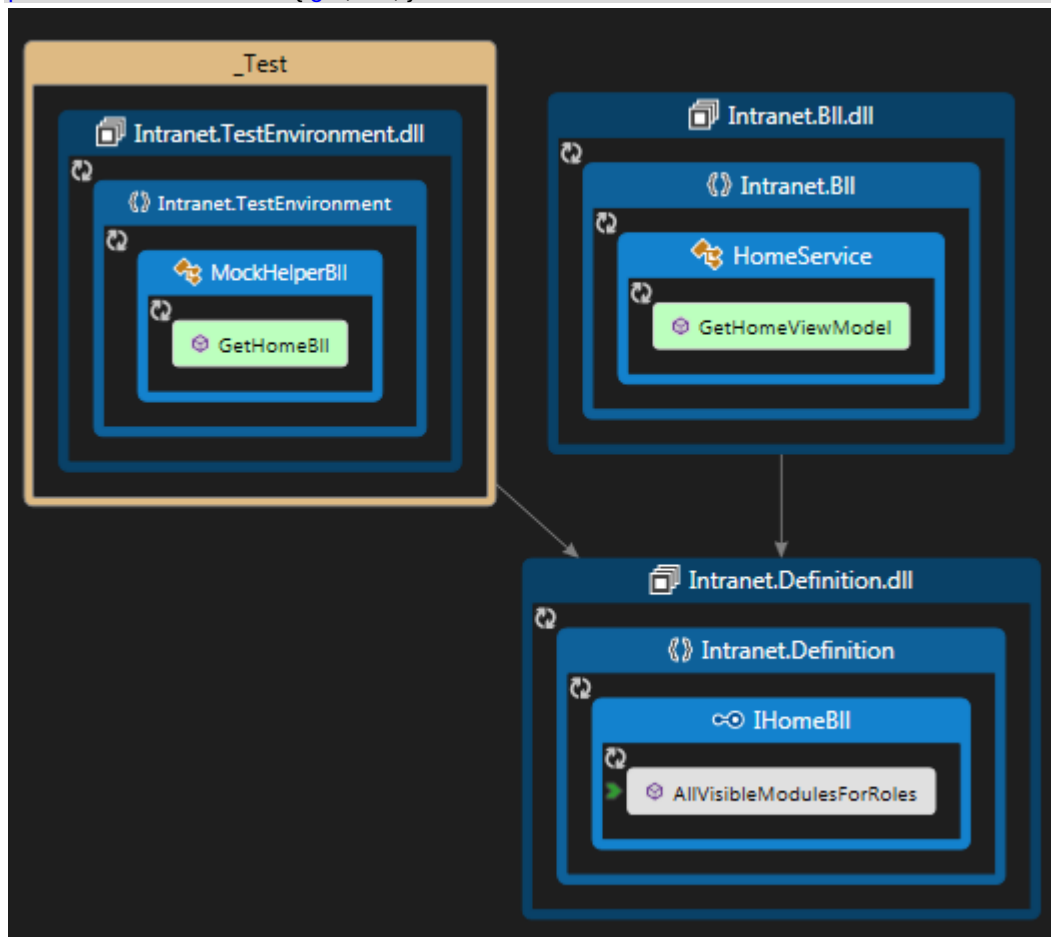


Abbildung 7: Code Map von Home Service via Visual Studio

Hier sieht man, dass Home Service sowie der Mock für den Bll Abhängigkeiten auf das Interface von HomeBll besitzen.

Wenn HomeBll das erste Mal benutzt wird, weiss Autofac wo es liegt, da wir es im Autofac Container registriert haben, und findet automatisch die registrierten Werte und injiziert es für uns.

Autofac unterstützt property, constructor parameter und method injection. Wir benutzen constant property injection da es für uns die eleganteste Lösung ist.

Da wir keine festen Abhängigkeiten besitzen, sondern nur Abhängigkeiten via Interfaces, vereinfacht dies uns das Unit Testen sehr. Wir mocken mit dem Framework Moq alle Abhängigkeiten.

Eine Erklärung zu Moq befindet sich im Kapitel 3.9.1.2.

3.2.3 MEF Evaluation

3.2.3.1 Was ist MEF

MEF (steht für Managed Extensibility Framework) ist eine Bibliothek für die Erstellung von erweiterbaren Anwendungen. Anwendungsentwickler können mit MEF Erweiterungen ohne Konfiguration ermitteln und verwenden. Erweiterungsentwickler ermöglicht MEF den Code leicht zu kapseln und Abhängigkeiten zu vermeiden. Mit MEF können die Erweiterungen nicht nur innerhalb der Anwendungen, sondern auch anwendungsübergreifend wiederverwendet werden.³

3.2.3.2 Unterschied zu IoC

Das Managed Extensibility Framework (MEF) und das inversion of control design Prinzip (IoC) sind von der Grundfunktionalität her ähnlich. Der Hauptunterschied zwischen ihnen ist, dass IoC Container angewendet werden mit statischen Abhängigkeiten, welche zur compile-time bekannt sind. Wo hingegen MEF bei dynamischen Abhängigkeiten, welche erst zur Laufzeit bekannt sind, angewendet wird.

MEF ist dafür optimiert, wenn man mit unbekannten Komponenten arbeitet, da man diese nicht registrieren muss.

3.2.3.3 Entscheidung für IoC

Nach längerem Austesten vom Managed Extensibility Framework (MEF), haben wir uns dazu entschieden in der Studienarbeit weiter einen IoC Container zu benutzen. Ein Hauptgrund dafür war, dass MEF benutzt wird, wenn es 3rd-Party Entwickler gibt welche Erweiterungen für das System schreiben was bei uns nicht der Fall ist. In dieser Studienarbeit wird die ganze Applikation von einer Gruppe von Entwicklern geschrieben, denen alles bekannt ist, wofür die Benutzung eines IoC Containers die wohlmöglichste Lösung ist. Die Intranet Applikation soll zwar mit Modulen/Plug-Ins erweiterbar sein, aber diese Erweiterungen werden nicht von 3rd-Party Entwicklern geschrieben. Ein weiterer Grund war auch, dass ein Grossteil unserer Architektur (die Shell) schon nach dem IoC-Prinzip aufgebaut war.

Wir haben auch noch die Möglichkeit in Betracht gezogen IoC mit MEF zu kombinieren, aber nach längerer Recherche haben wir diese Möglichkeit verworfen, da keine ausreichende Dokumentation dafür vorhanden war und die Zeit langsam knapp wurde.

³ Siehe [6]

3.3 Source Code

3.3.1 Solution / Projects

Es gibt eine Solution mit dem Namen "Intranet" und Projekte für jeden Layer.

3.3.1.1 Visual Studio

Solution 'Intranet'

- **_Test** Alle Testprojekte für die Shell (Solution Ordner)
 - Intranet.Bll.Test Test Projekt für Bll
 - Intranet.Common.Test Test Projekt für Common
 - Intranet.Dal.Test Test Projekt für Dal für die Shell
 - Intranet.Shell.Definition Definition beinhaltet alle Schnittstellen für die Shell
 - Intranet.Model.Test Test Projekt für Model
 - Intranet.Web.Test Test Projekt für Web
 - Intranet.ViewModel.Test Test Projekt für ViewModel
- **Labor** Alle Projekte zum Labor (Solution Ordner)
 - Intranet.Labor.Definition Definition beinhaltet alle Schnittstellen für das Labor
 - Intranet.Labor.Dal Data Access Layer ist zuständig für den Zugriff auf die Datenbank für das Labor
 - Intranet.Labor.Bll Die Business Logik für das Labor
 - Intranet.Labor.Model Die EF Models für das Labor
 - **_Test** Alle Testprojekte für das Labor (Solution Ordner)
 - Intranet.Labor.Dal.Test Test Projekt für Dal für das Labor
 - Intranet.Labor.Bll.Test Test Projekt für Bll für das Labor
- Intranet.Bll Business Logic Layer, beinhaltet die gesamte Logik
- Intranet.Common Common beinhaltet alles, welches über alle Projekte gebraucht wird. Common können Interfaces oder Basisklassen sein.
- Intranet.Dal Data Access Layer ist zuständig für den Zugriff auf die Datenbank für die Shell
- Intranet.Definitions Definitions beinhaltet alle Schnittstellen für die Shell
- Intranet.Model Model beinhaltet die EF Models
- Intranet.ViewModel ViewModel beinhaltet alle View Models
- Intranet.Web Web beinhaltet Controller und Views

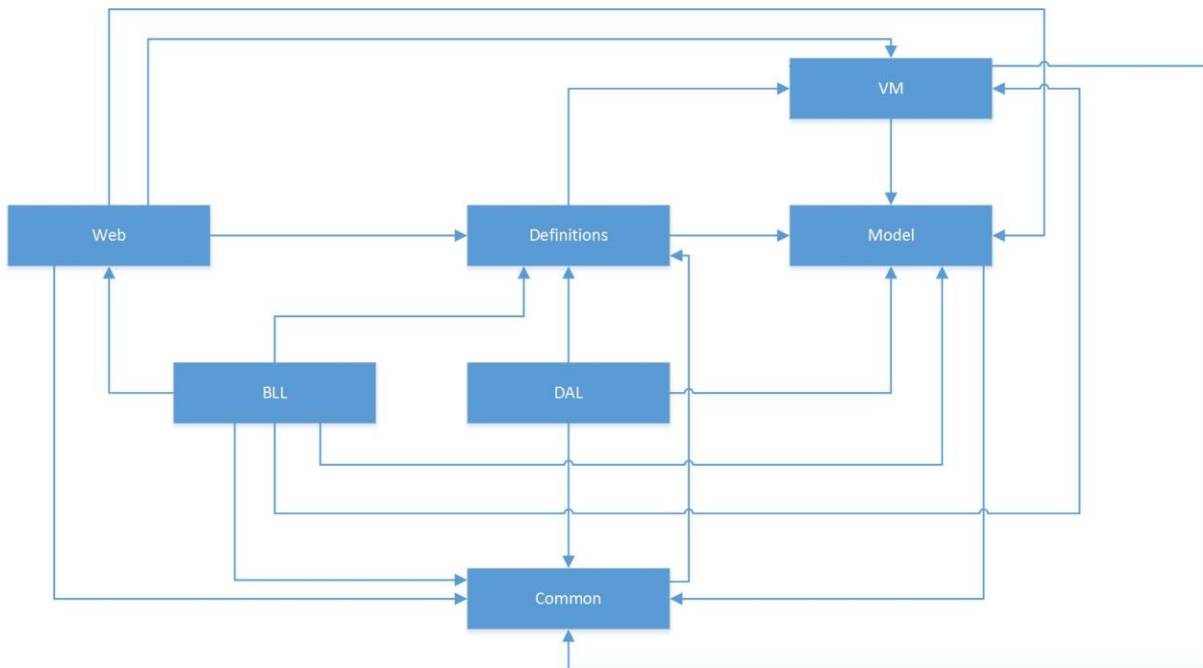


Abbildung 8: Abhängigkeiten

Abbildung 8 zeigt Grob auf wie die Abhängigkeiten aussehen. Für die Einfachheit haben wir uns auf die Shell fokussiert. Das Modul Labor wird für DAL/BLL/Definition im gleichen Konzept aufgebaut.

Definition wird vom Web, BLL, DAL und Common referenziert, da alle Schnittstellen (Interfaces) in Definition definiert werden.

Common beinhaltet gemeinsam benutzten Implementationen sowie Basisklassen.

Das Model wird von allen Projekten referenziert da es in den Interfaces vorkommt und für das Model keine Interfaces geschrieben werden, da die Modelklassen nur aus properties bestehen.

Das ViewModel wird von Definition für die Interfaces referenziert, und das Web benutzt die ViewModels des Projektes. Die BLL referenziert das Projekt, da auch für die ViewModels keine Interfaces bestehen, der Grund ist derselbe wie bei den Models.

3.3.1.2 Ordnerstruktur

Intranet

- .build Ablage für die Build Dateien
- .src Source Code
- .git automatisch erstellte Daten zu git
- .tools Tools welche im Projekt und im Build gebraucht werden
- .log Log Files
- .docs doc Projekt für die API Dokumentation Generierung
- Docs API Dokumentation
- appveyor.yml Appveyor Konfiguration File

.tools

- Cake Build Tool
- NuGet Für Packages
- XUnit Für Unit Tests
- Selenium Für Selenium Tests

.src

- | | |
|--------------------|---|
| Intranet.sln | Solution |
| - Intranet | Sourcen zu Intranet Solution ausser Test Projekte |
| - Common | Intranet.Common |
| - Definition | Intranet.Definition |
| - Dal | Intranet.Dal |
| - Bll | Intranet.Bll |
| - Model | Intranet.Model |
| - ViewModel | Intranet.ViewModel |
| - Web | Intranet.Web |
| - Labor | |
| ▪ Definition | Intranet.Labor.Definition |
| ▪ Dal | Intranet.Labor.Dal |
| ▪ Bll | Intranet.Labor.Bll |
| ▪ Model | Intranet.Labor.Model |
| ▪ ViewModel | Intranet.Labor.ViewModel |
| - Intranet.Testing | Alle Test Projekte |
| - Bll | Intranet.Bll.Test |
| - Dal | Intranet.Dal.Test |
| - Common | Intranet.Common.Test |
| - Model | Intranet.Model.Test |
| - ViewModel | Intranet.ViewModel.Test |
| - Web | Intranet.Web.Test |
| - TestEnvironment | Mocks für Tests für Intranet.X.Test |
| - Labor | |
| ▪ Bll | Intranet.Labor.Bll.Test |
| ▪ Dal | Intranet.Labor.Dal.Test |
| ▪ Model | Intranet.Labor.Model.Test |
| ▪ ViewModel | Intranet.Labor. ViewModel.Test |
| ▪ TestEnvironment | Mocks für Tests für Intranet.Labor.X.Test |

3.4 Schichtenmodell

3.4.1 Version 1 (Startidee)

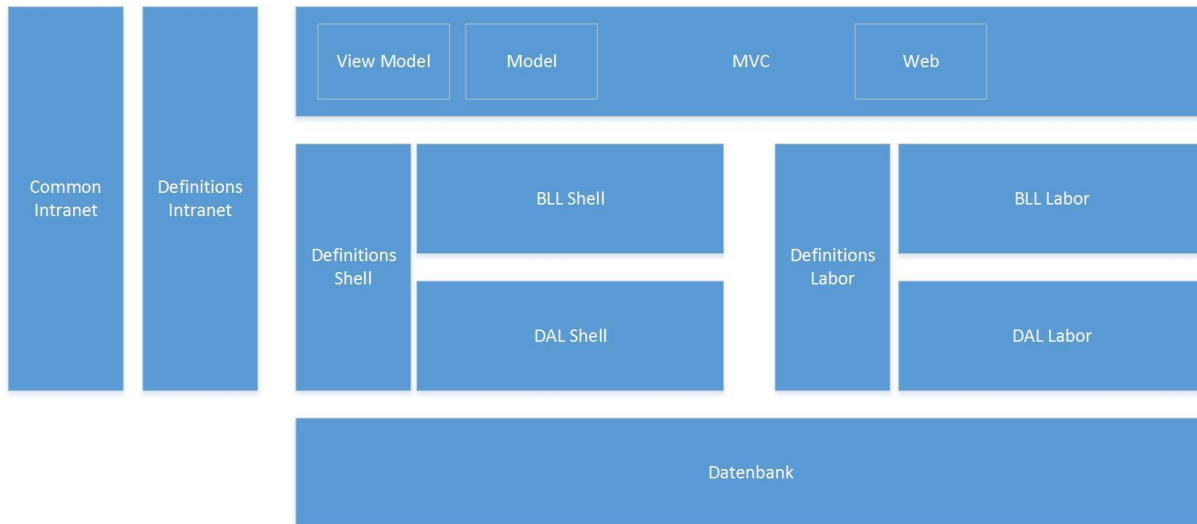


Abbildung 9: Schichtenmodell Version 1

Pro Modul gibt es ein Eigenes Projekt für den Layer BLL, DAL und Definitions.
 Im MVC gibt es pro Modul je eine eigene Area.
 Für die ganze Solution gibt es nur eine Datenbank.

3.4.2 Version 2

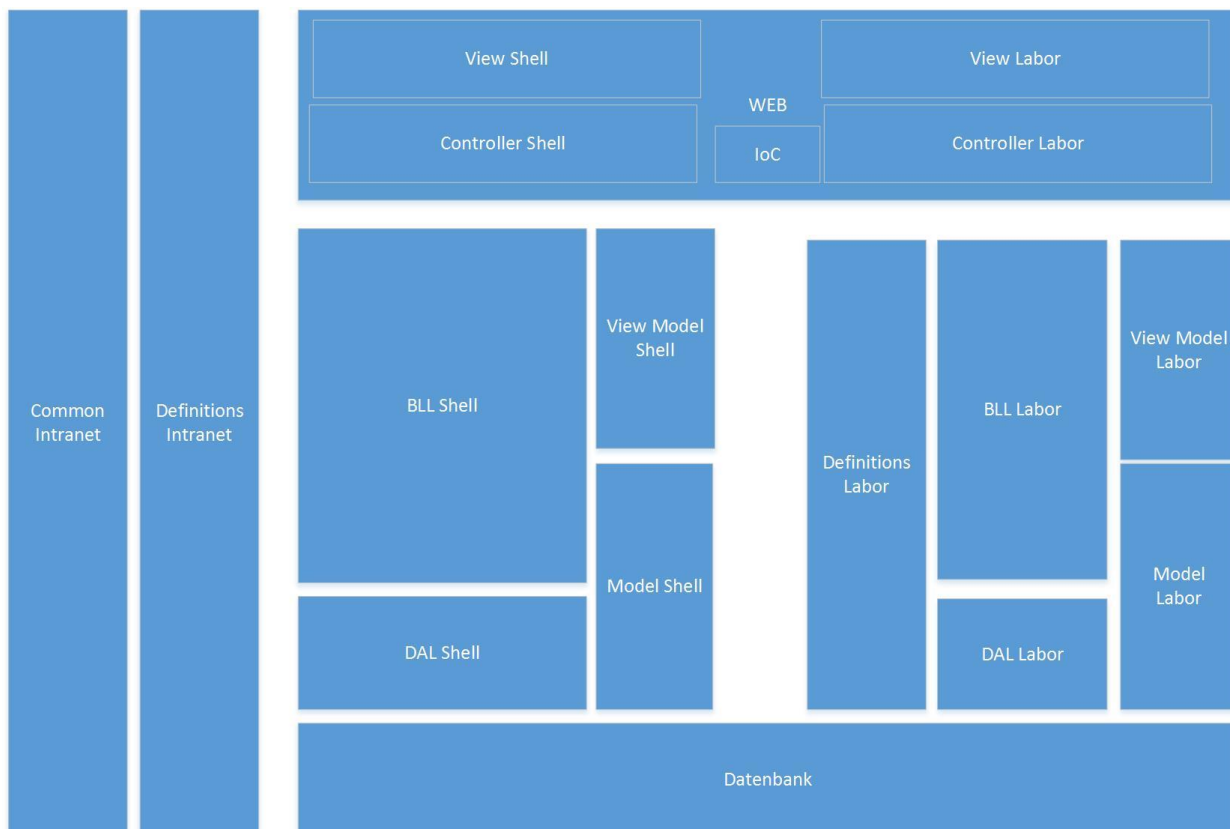


Abbildung 10: Schichtenmodell Version 2

Wir wollten die Shell und das Labor noch mehr trennen.

Deshalb gibt es für die Shell sowie das Labor nun ein eigenes Model und Viewmodel.
Zum Web gehört IoC.

3.4.3 Version 3

Wir waren nicht mit der Aufteilung von Common und Definition zufrieden.
Nun haben wir Definition so definiert, dass Definition nur die Shell Interfaces beinhaltet.
Das Common Projekt beinhaltet Schnittstellen, das heisst Interfaces und Basis Klassen, welche in der ganzen Solution verwendet werden

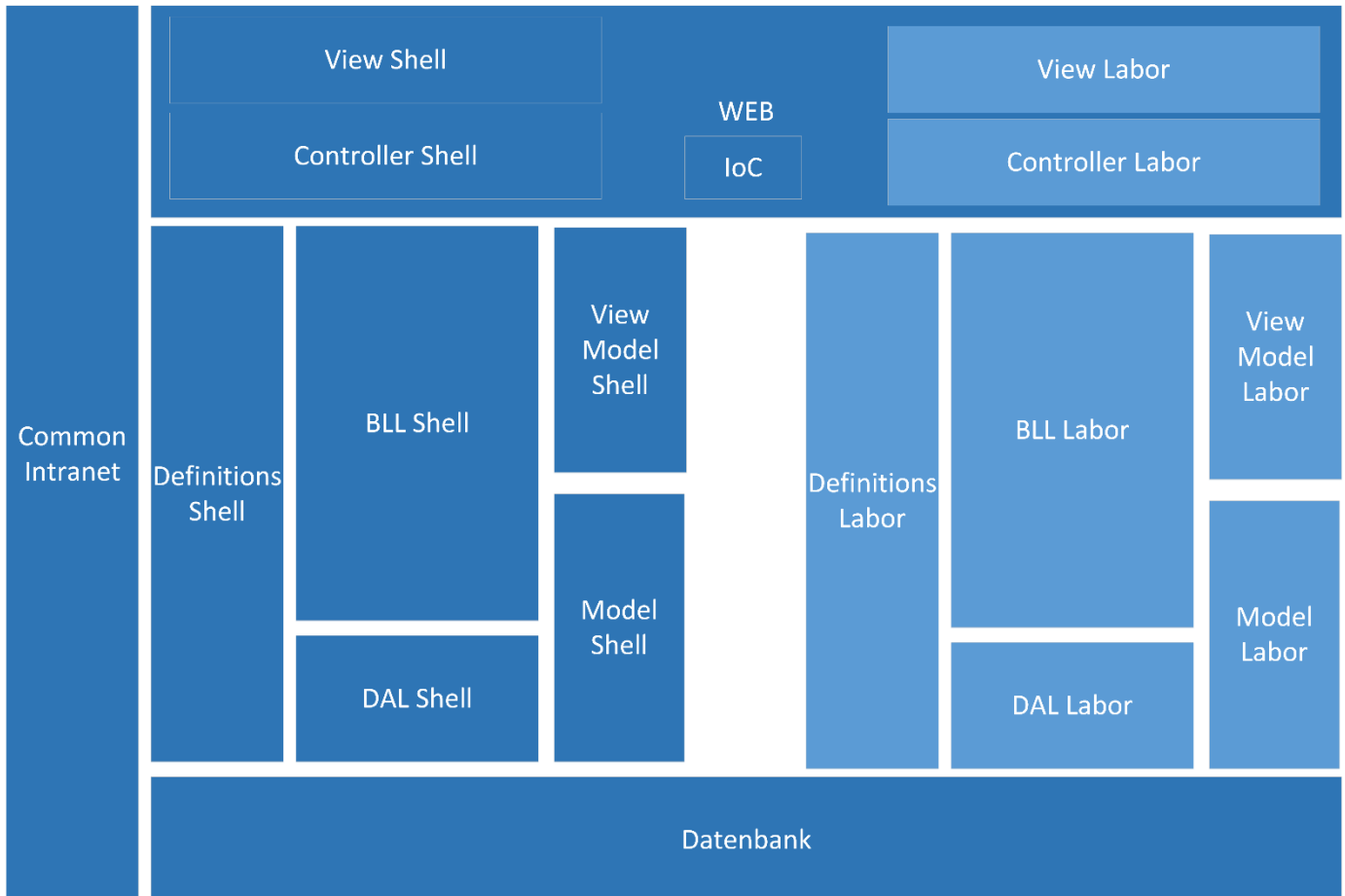


Abbildung 11: Schichtenmodell Version 3

3.5 Build

3.5.1 Appvoyer

Appvoyer ist ein Build server in der Cloud, welcher ein Continuous Delivery service für Windows bietet.
Das Projekt ist ersichtlich unter: <https://ci.appveyor.com/project/Saxer/intranet-lnd7y>
Appvoyer ist mit github verknüpft, und lässt nach jedem push auf github ein buildsript laufen.
Das Buildsript ist abgelegt unter .build\Build.ps1.
Die Konfigurationen für appvover sind im Root im Yml-File appveyor.yml abgelegt.

3.5.2 Cake

Das Build-Script build.ps1 im Ordner .build welches vom Appvoyer gestartet wird, startet das Build File welches mit Cake geschrieben ist.

Cake ist ein cross platform build automation system, welches Entwickler erlaubt das build script in C# zu schreiben.

Das Build Script build.cake ist in Tasks aufgebaut: Clean → RestorePackages → Build → RunTests.

- Clean → säubert die build output directories
- Restore Packages -> Wiederherstellt die Nuget Packete
- Build → Buildet die Solution mit MSBuild
- RunTests → Lässt alle Tests mit XUnit laufen

Nach dem der Build abgelaufen ist, sind die build Informationen (Build Console, Tests ...) auf Appvoyer ersichtlich.

3.5.3 Slack

Mit Slack als Projektchat zeigen wir auch alle commits auf Github und builds von Appvoyer an. Somit kann einfach nachvollzogen werden, was geschah.

3.6 Web Technology

Bei der Webseite verwenden wir unter anderem SASS und Typescript.

3.6.1 Gulp

Mit Gulp Tasks konvertieren wir SASS zu CSS, und komprimieren das CSS. TypeScript wird zu JavaScript und auch hier noch komprimiert. Mit Gulp watches machen wir uns das Entwickeln einfacher.

3.7 Routing

Wir benutzen in dieser Arbeit die Default Routing Implementation von ASP.NET. Dabei ist jede Route der Shell wie folgt definiert: `http(s)://{host}/{controller}/{action}/{id}`

- host: Domainname des Intranets (z.B. intranet.intigena.com)
- controller: Name des Controllers (z.B. Home)
- action: Der Actionname des Controllers (z.B. Index)
- id: Parameter, welcher der Action übergeben werden kann

Die Startseite des Intranets wäre somit `https://intranet.intigena.com/Home/Index`.

Gibt man nur `intranet.intigena.ch` ein landet man per Default auch auf der Startseite.

Für das Routing der Module wurden Areas benutzt. Jedes Modul besitzt seine eigene Area. Das Routing beim Modul sieht wie folgt aus: `http(s)://{host}/{area}/{controller}/{action}/{id}` wobei area der Name des Modules ist.

Die Folgende Tabelle (*Tabelle 1: Routing Übersicht*) gibt einen Überblick über alle Routes, die im Endergebnis der Studienarbeit vorhanden sind.

Route	Beschreibung
/Home/Index	Startseite des Intranets
/Settings/Index	Einstellungsseite für den Admin, in der er Module aktivieren bzw. deaktivieren kann
/Settings/Update	Post-Request welcher ein Modul aktiviert bzw. deaktiviert.
/Labor/LaborHome/Index	Startseite des Labor Modules
/Labor/LaborDashboard/Index	Dashboard Seite des Labors
/Labor/History/Index	History-Seite
/Labor/History/Search	Post-Request welcher eine Liste aller Testblättern zur gesuchten FA-Nr. liefert
/Labor/LaborCreator/Index	Seite um aktuelle Testblätter auszusuchen oder neue zu erstellen
/Labor/LaborCreator/Edit	Post-Request welcher ein Testblatt zur aktuellen Schicht mit gewünschter FA-Nr. erstellt, sofern diese im System vorhanden ist
/Labor/LaborCreatorBaby/Edit/{id}	Testblatt mit ID={id} für Babywindeln
/Labor/LaborCreatorInko/Edit/{id}	Testblatt mit ID={id} für Inkonsistenz Windeln
/Labor/BabyDiaperRetention/Create	Liefert ein neues Eingabeformular für Babywindeln Retention Test
/Labor/BabyDiaperRetention/Edit/{id}	Liefert ein mit einem Babywindeln-Retention Test (ID={id}) abgefülltes Formular zum Editieren eines Testes
/Labor/BabyDiaperRetention/Delete/{id}	Löscht den Baby-Retention Test mit ID={id}
/Labor/BabyDiaperRetention/Save	Post-Request der ein mitgeliefertes Baby-Retention ViewModel abspeichert (erstellt oder updatet)
/Labor/BabyDiaperRewet/Create	Liefert ein neues Eingabeformular für Babywindeln Rewet Test
/Labor/BabyDiaperRewet /Edit/{id}	Liefert ein mit einem Babywindeln-Rewet Test (ID={id}) abgefülltes Formular zum Editieren eines Testes
/Labor/BabyDiaperRewet /Delete/{id}	Löscht den Baby-Rewet Test mit ID={id}
/Labor/BabyDiaperRewet /Save	Post-Request der ein mitgeliefertes Baby-Rewet ViewModel abspeichert (erstellt oder updatet)
/Labor/InkoAquisition/Create	Liefert ein neues Eingabeformular für Inko-Aquisition Test
/Labor/InkoAquisition/Edit/{id}	Liefert ein mit einem Inko-Aquisition Test (ID={id}) abgefülltes Formular zum Editieren eines Testes
/Labor/InkoAquisition/Delete/{id}	Löscht den Inko-Aquisition Test mit ID={id}
/Labor/InkoAquisition/Save	Post-Request der ein mitgeliefertes Inko-Aquisition ViewModel abspeichert (erstellt oder updatet)
/Labor/InkoRetention/Create	Liefert ein neues Eingabeformular für Inko-Retention Test
/Labor/InkoRetention/Edit/{id}	Liefert ein mit einem Inko-Retention Test (ID={id}) abgefülltes Formular zum Editieren eines Testes
/Labor/InkoRetention/Delete/{id}	Löscht den Inko-Retention Test mit ID={id}

/Labor/InkoRetention/Save	Post-Request der ein mitgeliefertes Inko-Retention ViewModel abspeichert (erstellt oder updatet)
/Labor/InkoRewet/Create	Liefert ein neues Eingabeformular für Inko-Rewet Test
/Labor/InkoRewet /Edit/{id}	Liefert ein mit einem Inko-Rewet Test (ID={id}) abgefülltes Formular zum Editieren eines Testes
/Labor/InkoRewet/Delete/{id}	Löscht den Inko-Rewet Test mit ID={id}
/Labor/InkoRewet/Save	Post-Request der ein mitgeliefertes Inko-Rewet ViewModel abspeichert (erstellt oder updatet)

Tabelle 1: Routing Übersicht

3.8 Security

3.8.1 Authentifikation

Ein Benutzer wird durch ein Active Directory (kurz AD) authentifiziert. Ist ein Benutzer nicht via dem AD authentifiziert, so hat er zu keiner Funktion des Intranets Zugang.

3.8.2 Autorisation

Die Autorisation in der Intigena Intranet Applikation ist rollenbasiert. Dabei gibt es für jedes Modul eigene Rollen. Beim Labor-Modul gibt es zwei Rollen. Eine Rolle für Benutzer welche nur die Testdaten des Labors lesen kann und eine Rolle welche zusätzlich noch Testwerte erstellen, bearbeiten und löschen kann. Zusätzlich gibt es noch eine Administratorrolle, jene hat auf alle Module Zugriff sowie auf Admin-Tools (im Moment für die Aktivierung und Deaktivierung von Modulen).

Die Verschiedenen Rollen werden über jedem Controller im Code hinzugefügt für welche sie berechtigt sind.

```

[Authorize( Roles =
    RoleSettings.LaborAdmin + "," +
    RoleSettings.LaborUser + "," +
    RoleSettings.LaborViewer )]
3 references | Frank.Jordi, 6 days ago | 2 authors, 12 changes
public class LaborCreatorBabyController : BaseController
  
```

Abbildung 12: Rollenbeispiel für die Autorisation

Da die Rollennamen während der Entwicklung nicht unbedingt gleich heissen wie später im Einsatz, existiert eine RoleSettings-Klasse in der alle Rollen definiert sind damit sich die Namensänderungen der Rollen zentral an einem Ort befinden.

Zusätzlich zur der Controller-Autorisation gibt es noch eine Autorisation via der Datenbank, welche für die Sichtbarkeit der einzelnen Module verantwortlich ist.

3.9 Implementation (Entwicklung und Test)

3.9.1 Automatische Testverfahren

3.9.1.1 XUnit

Wir machen unsere Unit Tests mit XUnit.

XUnit ist ein open Source Test Tool für das .Net Framework ⁴

Mit [Fact] definiert man eine Methode als XUnit Test.

```
[Fact]
public void Test(){...}
```

Diese Tests kann man auf der Konsole oder direkt in Visual Studio laufen lassen.

Wir tun beides. In Visual Studio während der Entwicklung. Und via Konsole mit cake⁵ während der Entwicklung und danach.

3.9.1.2 Moq

Um korrekte Unit Tests ohne Abhängigkeiten durzuführen benutzen wir das Framework Moq.⁶

Wir mocken für unsere Unit Tests alle Abhängigkeiten. Wie bereits in einem vorherigen Kapitel erwähnt, ist dies sehr einfach für uns, da wir Dependency Injection durch unser gesamtes Projekt gezogen haben.

Ein sehr einfacher Moq kann etwa so aussehen:

```
public static IHomeBll GetHomeBll( IEnumerable<Module> modules )
{
    var mock = new Mock<IHomeBll>
    {
        Name = "MockHelper.GetHomeBll",
        DefaultValue = DefaultValue.Mock
    };

    mock.Setup( x => x.AllVisibleModulesForRoles( It.IsAny<IEnumerable<String>>() ) )
        .Returns( modules );

    return mock.Object;
}
```

Wir erstellen einen Mock für das Interface IHomeBll, welches einen Namen und ein Default Wert besitzt. Danach registrieren wir einen Mock für die Methode AllVisibleModulesForRoles, welcher für jeden Input type von IEnumerable<String> die Module zurückgibt, welche man bei dem Aufruf des Mocks mitgibt.

⁴ Die Offizielle Webseite zu XUnit [8]

⁵ Siehe Kapitel 3.5.2

⁶ Webseite und Erklärung von Moq von den Entwicklern [9]

Eine einfache Verwendung eines solchen Mocks in Verbindung mit einem XUnit Tests kann so aussehen:

```

/// <summary>
///   Normal Passing Test with no value
/// </summary>
[Fact]
public void GetHomeViewModelEmptyTest()
{
  ①   var homeBllMock =
      MockHelperBll.GetHomeBll(
          new List<Module>()
      );

  ②   var rolesMock = MockHelperRoles.GetRoles( new List<String>() );

      var target = new HomeService( new NLogLoggerFactory() )
      {
  ③       HomeBll = homeBllMock,
          Roles = rolesMock
      };

  ④   var actual = target.GetHomeViewModel();

      actual.Modules.ToList().Count.Should().Be( 0 );
}
  
```

- ① Wir erstellen einen neuen Mock für HomeBll und geben eine Liste von Modulen als Input mit, jene werden, wie man im mock Beispiel sehen kann, für die Methode AllVisibleModulesForRoles zurückgegeben.
- ② Wir erstellen einen Mock für die Rollen, welches ein weiteres Interface IRoles mockt.
- ③ Wir erstellen das Objekt HomeService welches wir in diesem Beispiel testen wollen. Da HomeBll und Roles properties sind, welche über Autofac automatisch erstellt werden, und wir dies im Test nicht mit Autofac machen, setzten wir nun für sie unsere passenden mocks.
- ④ Beim Aufruf von GetHomeViewModel wird jene Methode ausgeführt

```

/// <summary>
///   All main models and setting models that the current User is allowed to see.
/// </summary>
/// <returns>The ViewModel for the home</returns>
public HomeViewModel GetHoViewModel()
{
    var roleNames = Roles.GetRolesForUser();

    var vm = new HomeViewModel
    {
        Modules = HomeBll.AllVisibleModulesForRoles( roleNames )
    };
    return vm;
}
  
```

Wie man sieht wird hier von HomeBll die Methode AllVisibleModulesForRoles aufgeführt, welche wir gemockt haben. Somit beinhaltet unser ViewModel danach die Werte welche wir beim Erstellen des Mocks für HomeBll definiert haben. In unserem Fall eine Leere Liste mit dem Typ Module.

3.9.1.3 Tests auf dem Server

Auf dem Server werden alle Tests mit Cake⁷ ausgeführt.

Das Code Snippet zeigt den Task RunTests aus dem File build.cake welches sich im .build Ordner befindet.

```
// Run the unit tests
Task("RunTests")
    .IsDependentOn("Build")
    .Does(() =>
    {
        XUnit2(new []{
            testDirectory.ToString()+"/Web/bin/Release/Intranet.Web.Test.dll",
            testDirectory.ToString()+"/Bll/bin/Release/Intranet.Bll.Test.dll",
            testDirectory.ToString()+"/Common/bin/Release/Intranet.Common.Test.dll",
            testDirectory.ToString()+"/Dal/bin/Release/Intranet.Dal.Test.dll",
            testDirectory.ToString()+"/Labor/Bll/bin/Release/Intranet.Labor.Bll.Test.dll",
            testDirectory.ToString()+"/Labor/Dal/bin/Release/Intranet.Labor.Dal.Test.dll",
            testDirectory.ToString()+"/Labor/Model/bin/Release/Intranet.Labor.Model.Test.dll",
            testDirectory.ToString()+"/Labor/ViewModel/bin/Release/Intranet.Labor.ViewModel.Test.dll",
            testDirectory.ToString()+"/ViewModel/bin/Release/Intranet.ViewModel.Test.dll"
        }, new XUnit2Settings
        {
            ToolPath = xUnit,
            HtmlReport = true,
            NoAppDomain = true,
            OutputDirectory = outputDirectory
        });
    });
```

Die Tests werden mit Cake laufen gelassen. Mit dem ToolPath wird angegeben wo das Tool abgelegt wird. Weiter oben im File steht:

```
var xUnit = toolDirectory + File("XUnit/xunit.console.exe");
```

Da der Task abhängig von Build ist, wird zuerst der Build ausgeführt. (Clean→Restore Packages → Build) Danach werden die Tests mit XUnit2, ausgeführt.

3.9.1.4 Controller Testing

Zu Beginn des Projektes haben wir begonnen die Controller mit Unit Tests zu testen. Im Laufe der Entwicklung sind wir aber zum Schluss gekommen, dass dies nicht sonderlich viel bringt. Die Controller besitzen keine grosse Logik die man hätte Testen können. Sie holen sich nur ein ViewModel vom Service und übergeben dies der View. Das einzige was in diesem Fall testbar wäre, ist, ob das in die View Abgefüllte ViewModel dasjenige ist, welches der View übergeben wurde, oder ob die richtige View aufgerufen wurde. Dies sind Funktionen, welche vom ASP.NET MVC Framework kommen und dessen Funktionalität wir deshalb nicht testen müssen.

⁷ Siehe Kapitel: 3.5.2

3.9.2 Manuelle Testverfahren

Der Systemtest wird in dieser Studienarbeit manuell durchgeführt. Dafür wurde ein Testplan erstellt, welcher einzelne Tests beinhaltet, die folgende Struktur haben:

Test X	
Funktionalität	
Vorbedingung	
Abhängigkeiten	
Ablauf	
Erwartete Resultate	

- Jeder Test hat eine Nummer
- Beschreibt welche Funktionalität dieser testet
- Was die Vorbedingungen für diesen Test sind
- Zu welchen anderen Tests dieser Abhängig ist
- Wie der Testablauf aussieht
- Was das erwartete Resultat ist

Für die Durchführung des Systemtestes gibt es noch eine Tabelle, in welcher eingetragen wird, ob die einzelnen Tests erfolgreich waren und im Falle eines Fehlers, was fehlgelaufen ist.

Anfänglich hatten wir vor unseren Systemtest, in einem automatischen Testverfahren mit Integrationstests, durchzuführen. Wir haben uns aber im Verlauf des Projektes dagegen entschieden. Einerseits aus zeitlichen Gründen da Integrationstests schreiben viel Zeit in Anspruch nimmt. Andererseits, weil ASP.NET Applikationen mühsam sind um Mittels Integrationstest zu testen. Auch könnten wir die Integrationstests nicht auf unserem Build Server testen, denn dazu würde dieser einen IIS brauchen um die Applikation zu starten.

Für die Evaluation der automatisierten Integrationstests wurden im Projekt aber Integrationstests für die Shell erstellt, welche aber beim Testen zurzeit ignoriert werden. Diese Integrationstests laufen mit dem Testframework Selenium und würden bei jedem einzelnen Test eine neue Instanz auf dem IIS starten.

4 Shell

4.1 Einleitung

Die Shell ist das zentrale Element der Intigena Intranet Applikation. Sie ist die Rahmen-Applikation der Intranet Applikation und kann durch Module, wie das entwickelte Labor-Modul, erweitert werden.

4.1.1 Admin Tools

Für die Intranet-Applikation gibt es eine Einstellungsseite auf welche nur der Admin Zugriff hat. Dort kann ein Administrator einzelne Module deaktivieren beispielsweise wieder aktivieren. Ob ein Modul Aktiviert oder Deaktiviert ist, wird in der Datenbank hinterlegt.

Greift ein Benutzer auf ein deaktiviertes Modul zu, so erscheint eine Error-Meldung in der ein Benutzer informiert wird, dass dieses Modul zurzeit nicht zur Verfügung steht und er sich an den Administrator wenden soll. Dies wird mittels eines selbst erstellten ActionFilterAttribute namens CheckDisable gelöst. Jeder Controller eines Moduls braucht hierzu ein Attribut, in welchem steht zu welchem Modul dieser gehört.

```
[CheckDisable(ModuleName = "Labor")]  
13 references | Frank Jordi, 6 days ago | 2 authors, 9 changes  
public class BabyDiaperRetentionController : BaseController
```

Abbildung 13: Labor Controller mit Attribut

Vor jedem Zugriff auf den Controller wird überprüft, ob dieses Modul Aktiviert ist und lässt den Benutzer dementsprechend entweder durch oder leitet ihn an eine Error-Seite weiter.

Der Zugriff beinhaltet eine Datenbankabfrage. Es hätte noch die Möglichkeit gegeben diese Information zu cachten. Wir entschieden uns aber für eine Datenbankabfrage, damit unter Umständen bei einem Einsatz von mehreren Servern der Cache nicht synchronisiert werden muss, sondern nur die Datenbank.

4.2 Datenbank

4.2.1 Datenbank Version 1

Wir haben ein Model für die Shell designt.

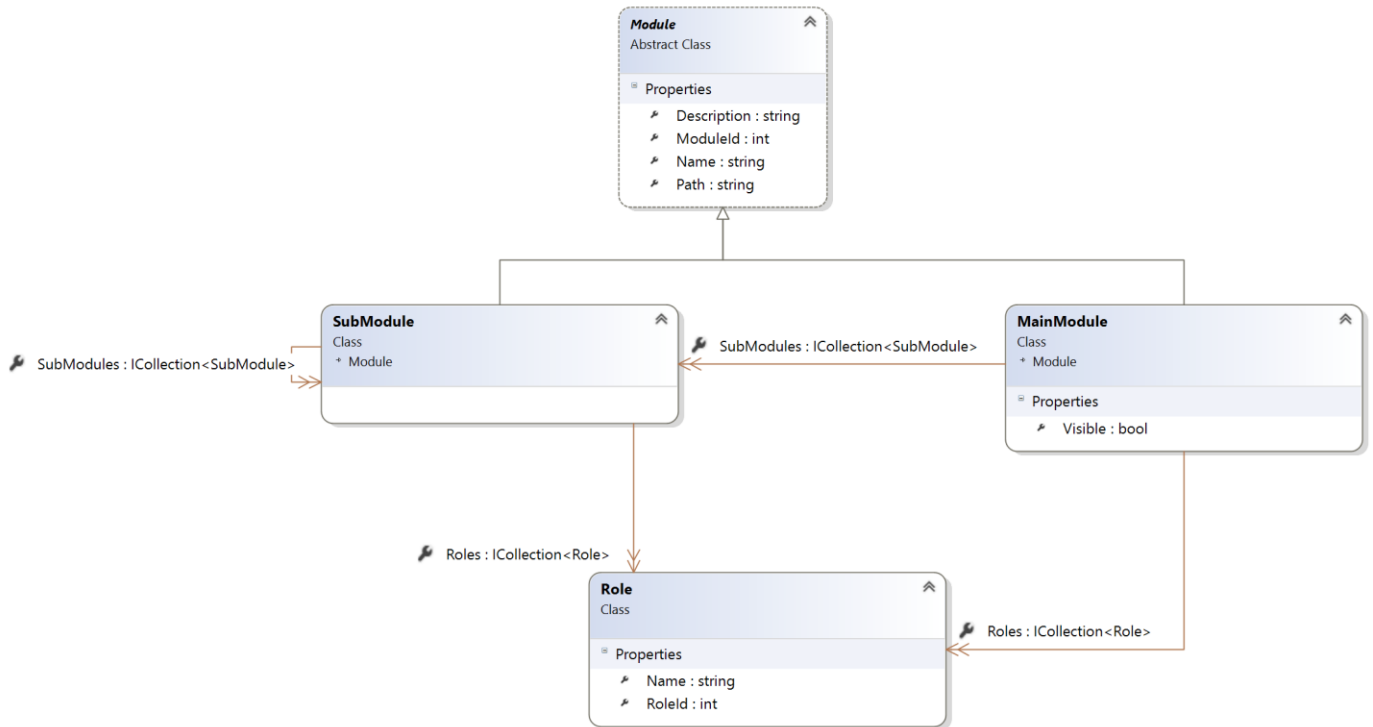


Abbildung 14: Klassendiagramm Shell Version 1

Die Shell besitzt Module.

Ein Modul ist entweder ein Main Modul oder ein SubModul.

Ein MainModul ist bei uns das Labor und kann vom Administrator aktiviert und deaktiviert werden.

Das MainModul kann beliebig viele SubModule besitzen.

Ein Submodul kann zum Beispiel das Labor Dashboard sein.

Die Module besitzen Rollen, welche bestimmen, wer das MainModul und die Submodule in der View sehen kann.

Wieso nicht das Modul beliebig viele Rollen besitzt, sondern die erbenden Klassen, ist ein Designentscheid der uns das EF vorgegeben hat, genaueres folgt noch.

4.2.1.1 TBH versus TBT versus TBC

Es gibt drei Möglichkeiten wie wir Vererbung darstellen können. Diese sind, Table per Hierarchie, Table per Type und Table per Concrete Class.⁸

⁸ Informationen von Quellen: [2] [3] [4] [1]

4.2.1.1.1 Table per Hierarchie

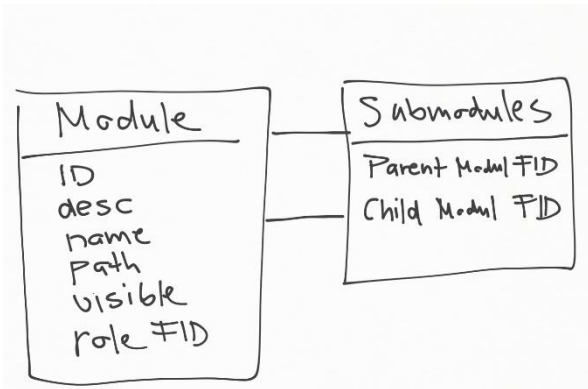


Abbildung 15: TBH Skizze

Table per Hierarchie hat eine Tabelle mit allen Informationen. Die Felder welche nicht gebraucht werden sind null.

Evaluierung

Table per Hierarchie hat den Vorteil, dass es eine hohe Performance bei CRUD Operationen hat, da alle Daten in einer Tabelle gespeichert werden. Und dass in der Datenbank eine minimale Anzahl von Tabellen existiert

Der Nachteil jedoch ist die Daten Redundanz, da viele null Werte existieren. Somit könnte auch manuellem ändern der Daten die Daten Integrität verletzt werden. Auch erschwert es das Hinzufügen und Löschen von Entitäten.

Da es mit TBH einfach ist die Datenintegrität zu verletzen und somit die dritte Normalform, kommt für uns diese Option nicht in Frage. Auch finden wir die Daten Redundanz sehr unelegant.

4.2.1.1.2 Table Per Type

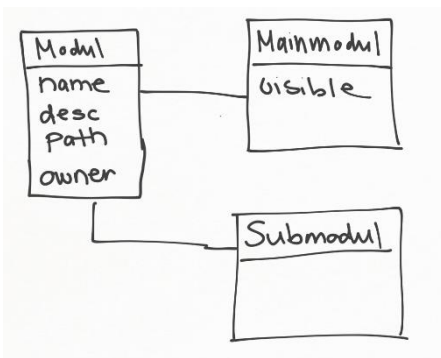


Abbildung 16: TBT Skizze

Table per Type hat eine Tabelle für die Basisklasse Modul und ein Feld owner, bei welcher ein String gespeichert ist, hier wäre es entweder "Mainmodul" oder "Submodul". Das EF löst dies dann automatisch auf.

Evaluierung

Der Vorteil ist, dass nun im Gegensatz von TBH, keine Daten Redundanz besteht. Auch ist nun Daten Integrität sichergestellt.

Der Nachteil jedoch ist, dass nun die CRUD Operationen langsamer werden. Vor allem wenn die Hierarchie noch wächst. Besonders schön finden wir dies nicht, aber besser als per Hierarchie.

4.2.1.1.3 Table Per Concrete Class

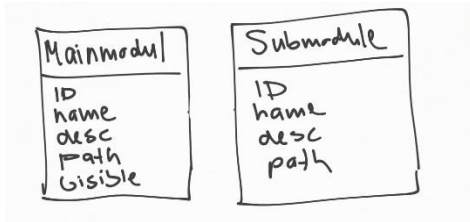


Abbildung 17: TPC Skizze

Pro Klasse, bei uns Mainmodul und Submodul, wird eine eigene Tabelle erstellt, mit allen Attributen, welche die konkrete Klasse braucht.

Evaluierung

Der Vorteil ist, dass im Gegensatz zu TBT die CRUD Operationen performanter sind und keine unnötige Joins gebraucht werden.

Es besteht keine Datenredundanz und auch keine Verletzung der Integrität ist möglich.

Für uns ist TPC die beste Option.

Der einzige Nachteil ist, dass das EF Beziehungen nicht auf dem BaseModule auflösen kann, da bei Role für das SubModule und MainModul je die Beziehung definiert werden muss.

4.2.1.1.4 Datenbankdiagramm

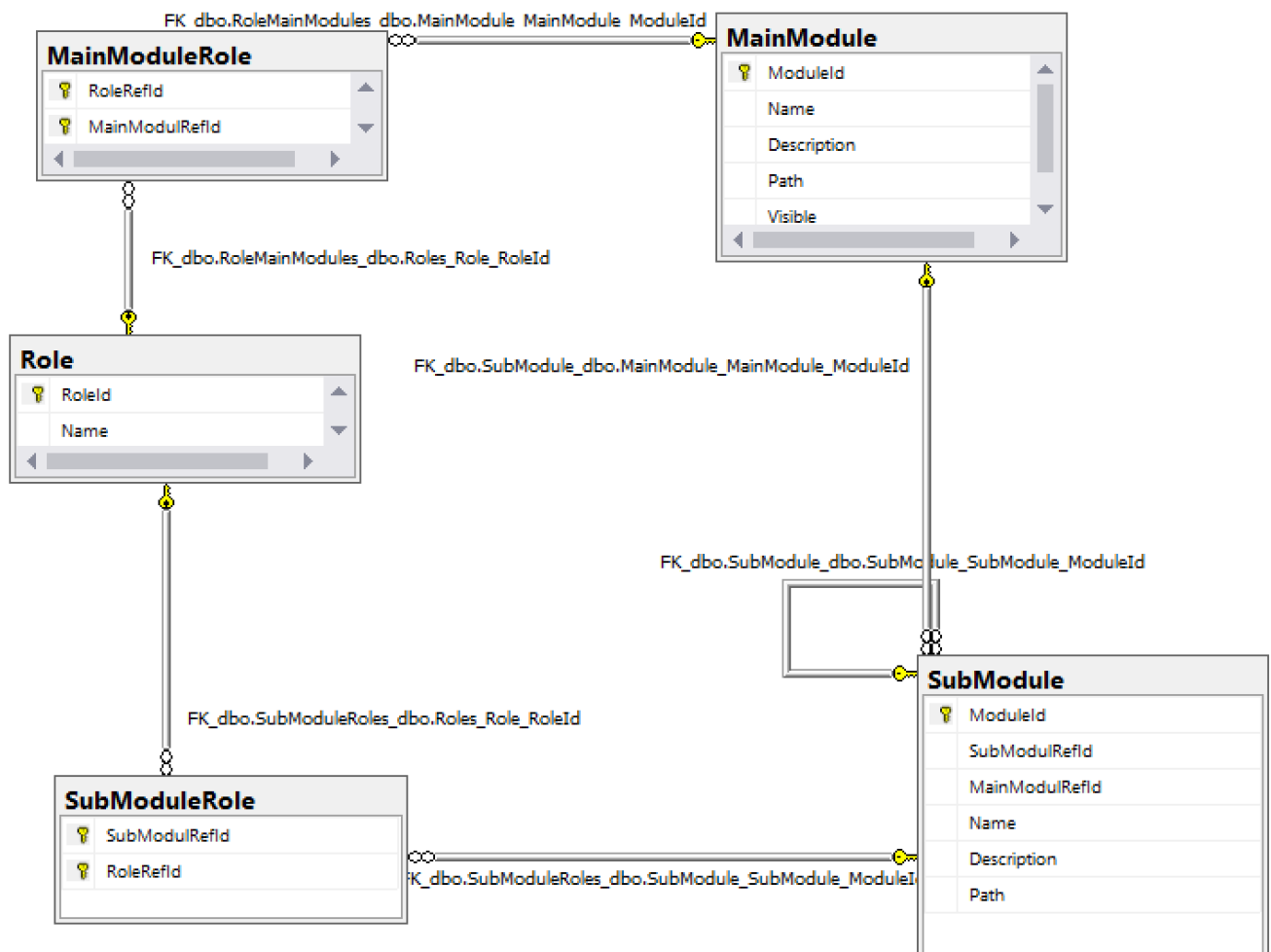


Abbildung 18: DB Diagramm Shell Version 1

4.2.2 Datenbank Version 2

Nach Code Review und weiterer Evaluation haben wir nun einige Nachteile unseres jetzigen Datenbankmodells festgestellt. Jedoch nicht konkret gegen die Entscheidungen welche wir bei der Evaluation von „TBH versus TBT“ versus TBC getroffen haben, sondern bei der grundsätzlichen Entscheidung unser Model mit Vererbung zu verwenden. Eine Datenbank sollte bei dem Konzept nicht mit der schlussendlichen Implementation Designt werden. Zuerst sollte sauber eine Datenbank erstellt werden ohne an die weitere Verwendung und den Einsatz zu denken.

Die Nachteile unserer Datenbank mit Vererbung sind:

- Für die Abfrage der Module muss man auf zwei Tabellen zugreifen, welche den Zugriff langsamer macht, als wenn es nur eine wäre.
- Die Abfragen werden komplizierter.
- Es bestehen mehr Tabellen als nötig, welche es unübersichtlich machen.

Deshalb haben wir unsere Datenbank von Grund auf neu überdacht und erstellt, diesmal ohne Vererbung. Hier entscheidet ein Typ was für eine Art von Module es ist.

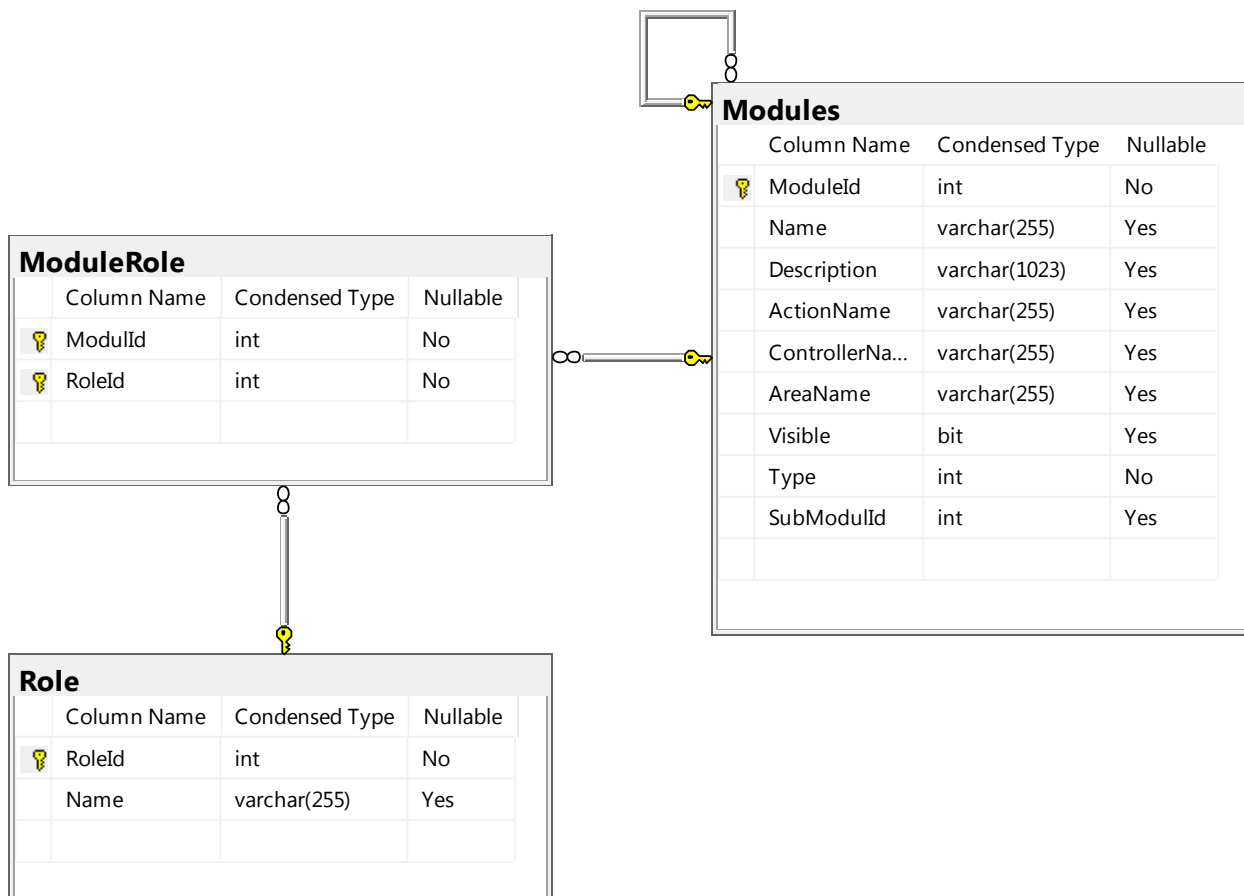


Abbildung 19: DB Diagramm Shell Version 2

Modules

Modules ist die Tabelle welche alle Module für die Intranet Seite beinhaltet. In der nachfolgenden Abbildung sind die Implementierten Module sichtbar.

	ModuleId	Name	Description	ActionName	ControllerName	AreaName	Visible	Type	SubModulRefId
1	1	Labor	Labor QS	Index	LaborHome	Labor	1	0	NULL
2	2	Creator	Labor QS Creator	Index	LaborCreator	Labor	NULL	1	1
3	3	Dashboard	Labor QS Dashboard	Index	LaborDashboard	Labor	NULL	1	1
4	4	Einstellungen	Einstellungen für die Shell	Index	Settings		1	2	NULL

Abbildung 20: DB Daten Shell Modules

Name	ist der Name des Modules welcher auch in der View angezeigt wird.
Description	ist die Modulbeschreibung.
ActionName	ist der Name der Action welche für dieses Modul ausgeführt wird.
ControllerName	ist der Name des Controllers welcher für das Modul zuständig ist.
AreaName	ist die Area des Modules. Ist es ein Main Modul, besitzt es keine Area.
Visible	wird für die Mainmodule gesetzt. Wenn 1 dann ist das Module sichtbar. Wenn 0 wurde das Modul deaktiviert.
Type	steht für den Typ des Modules. 0 bedeutet es ist ein Mainmodul. 1 bedeutet es ist ein Submodul. 2 bedeutet es ist ein Modul für die Settings.
SubModulRefId	Wird für Submodule gesetzt, somit ist klar welches Submodul zu welchem Mainmodul gehört.

Tabelle 2: DB Shell Werte Beschreibung

Roles

	RoleId	Name
1	1	Administrator
2	2	LaborUser
3	3	LaborViewer

Name Der Name der Rolle welcher auch so im Active Directory eingetragen ist.

Diese Namen sollen bei der Inbetriebnahme auf die gewünschten Rollen Namen angepasst werden.

- Administrator (ID 1) besitzt alle Rechte
- LaborUser (ID 2) besitzt alle Rechte auf das Labor
- LaborViewer (ID 3) besitzt View Rechte auf das Labor

Abbildung 21: DB Daten Shell Roles

Tabelle 3: DB Daten Shell Roles Werte Beschreibung

ModuleRoles

Ist die Zwischentabelle von Roles und Modules.

4.2.3 Generic Repository

Das Generic Repository repräsentiert ein Repository und ist eine Abstrakte Klasse welches das Interface IGenericRepository implementiert.

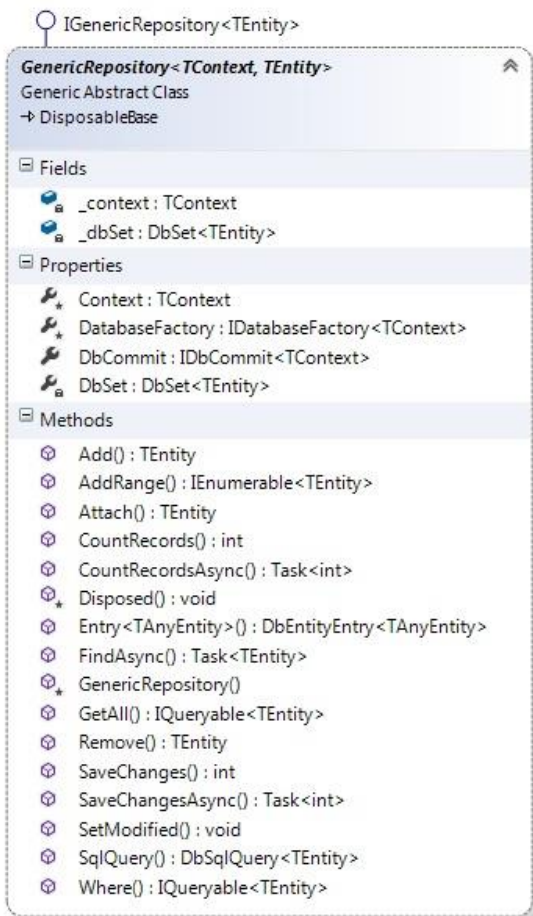


Abbildung 22: Klassendiagramm GenericRepository

Das Generic Repository beinhaltet die Implementationen welches jedes Repository braucht, damit man einfach Datenbankzugriffe machen kann.

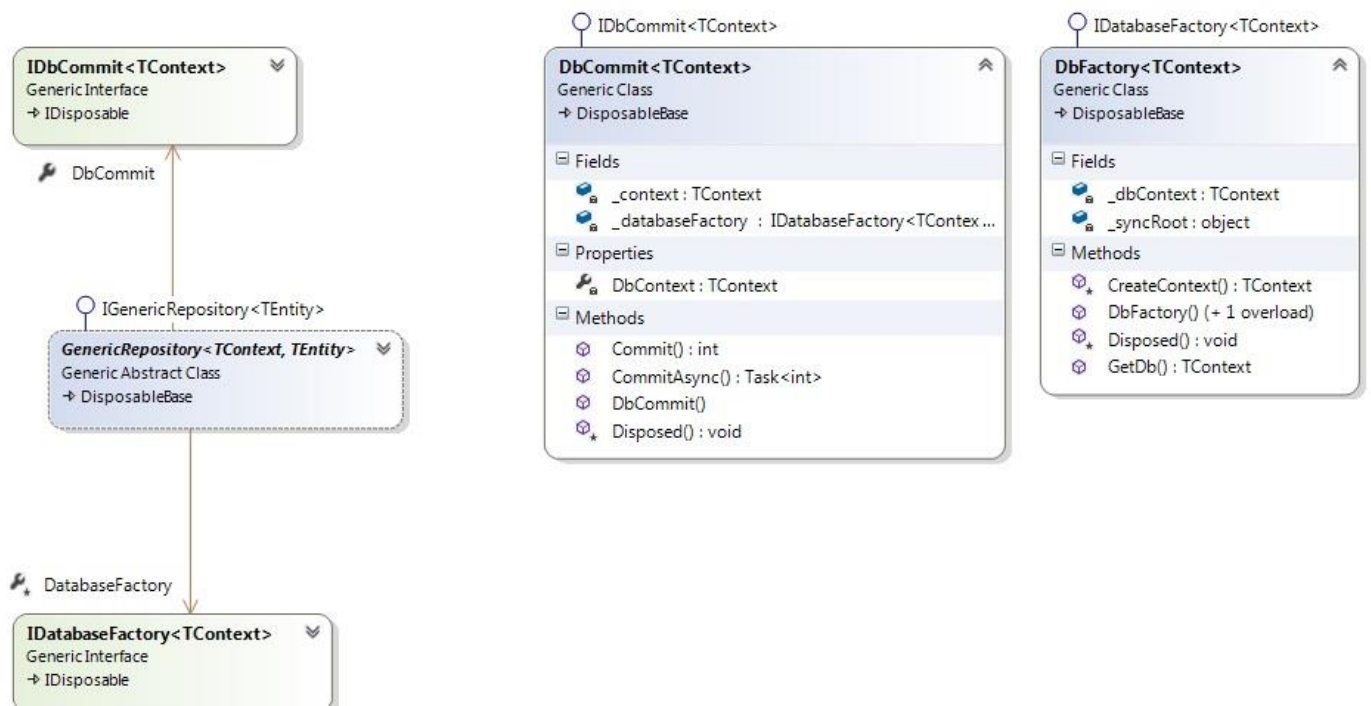


Abbildung 23: Klassendiagramm GenericRepository 2

Das GenericRepository hat unter anderem die Properties DbCommit und DbFactory.
DbCommit repräsentiert eine Klasse, welche fähig ist, Änderungen auf die Datenbank zu committen.
DbFactory ist die Klasse, welche den Datenbank Context erstellt.

Eine Verwendung im Data Access Layer kann somit so aussehen:

```
public class ModuleRepository : GenericRepository<IntranetContext, Module>
{
    public ModuleRepository( IDatabaseFactory<IntranetContext>
        databaseFactory, ILoggerFactory loggerFactory )
        : base( databaseFactory, loggerFactory )
    {
    }
}
```

Welches das Wiederverwenden von Code erspart.

Ein solches Repository wird danach im Business Logic Layer verwendet, um auf die Daten zuzugreifen.

```
public IGenericRepository<Module> ModuleRepository { get; set; }

public IEnumerable<Module> AllVisibleMainModules()
{
    var modules = ModuleRepository.GetAll()
        .Where( module => module.Type == ModuleType.Main )
        .ToList();
    return modules;
}
```

Da das Repository somit mit einem Interface abstrahiert wurde, kann es auch einfach für Unit Tests gemockt werden. Siehe Kapitel Moq: 3.9.1.2

Damit das richtige Repository mit IOC während der Laufzeit gefunden wird, wird es auch im Container registriert. Siehe Kapitel IOC: 0.

```
builder.RegisterAssemblyTypes( typeof(ModuleRepository).Assembly )
    .Where( t => t.Name.EndsWith( "Repository", StringComparison.Ordinal ) )
    .AsImplementedInterfaces()
    .PropertiesAutowired()
    .InstancePerRequest();
```

4.3 Model

Wir arbeiten mit EF Code First, unser Model widerspiegelt somit die Datenbank.
Die Klassendiagramme sind Kapitel 10.5.6 abgelegt.

4.4 Package Diagramm

Ein Package Diagramm über die Shell ist im Kapitel 10.4.1 abgelegt.

4.5 Navigation

Die Navigation wird dynamisch aufgebaut.

4.5.1 View

Die View der Navigation ist eine Partial View welche im `_Layout.cshtml` File gerendered wird.

```
@{ Html.RenderPartial( "_HeaderNavBar" ); }
```

Die PartialView greift auf den ViewBag zu, welcher alle Module zurückgibt, welche dargestellt werden sollten.

```
@foreach ( var item in ViewBag.Navigation.MainModules )
{
    <li>@Html.ActionLink((string)item.Name, (string)item.ActionName,
        (string)item.ControllerName, new { area = (string)item.AreaName },
        new { id = item.Name, @class = "intigena-nav-text" })
    </li>
}
```

4.5.2 Controller

Ausser bei der Navigation und dem Titel passen wir die Daten via View Object zur View da dies einiges eleganter ist als per ViewBag.

Hier das Beispiel für die Navigation mit dem unüblichen Übergeben der Daten via ViewBag.

Damit wir nicht in jedem Controller die gleichen Daten abfüllen müssen, haben wir einen BaseController, welcher uns das ViewModel in den ViewBag.Navigation speichert. Jeder unserer Controller erbt vom BaseController.

```
protected override void OnActionExecuting( ActionExecutingContext filterContext )
{
    ViewBag.Navigation = NavigationService.GetNavigationViewModel();
    base.OnActionExecuting( filterContext );
}
```

4.5.3 Service

Das ViewModel wird im Service abgefüllt. Unser NavigationService implementiert, wie alle unsere Services die abstrakte Klasse Service Base, welche ihm unter anderem die Funktionalität implementiert, alle Rollen des aktuellen Users zurückzugeben.

Unser NavigationService holt sich alle Rollen vom aktuellen User, ruft die NavigationBll mit den Userrollen als Attribut auf, und bekommt von der NavigationBll alle Module für den aktuellen User zurück.

4.5.4 Bll

Unsere Navigation Bll besitzt ein Repository mit dem Typ `IGenericRepository<Role>`, mit welchem er auf die Role Tabelle zugreift. Siehe Kapitel GenericRepository 4.2.3 und Tabelle Role 4.2.2 .

4.5.5 Diagramm

Zur Veranschaulichung über die beschriebenen Klassen, folgend noch das Diagramm zur Navigation.

Die Blau markierten Klassen sind im Projekt Intranet.Common abgelegt.

Die Übrigen im Projekt Intranet.Bll.

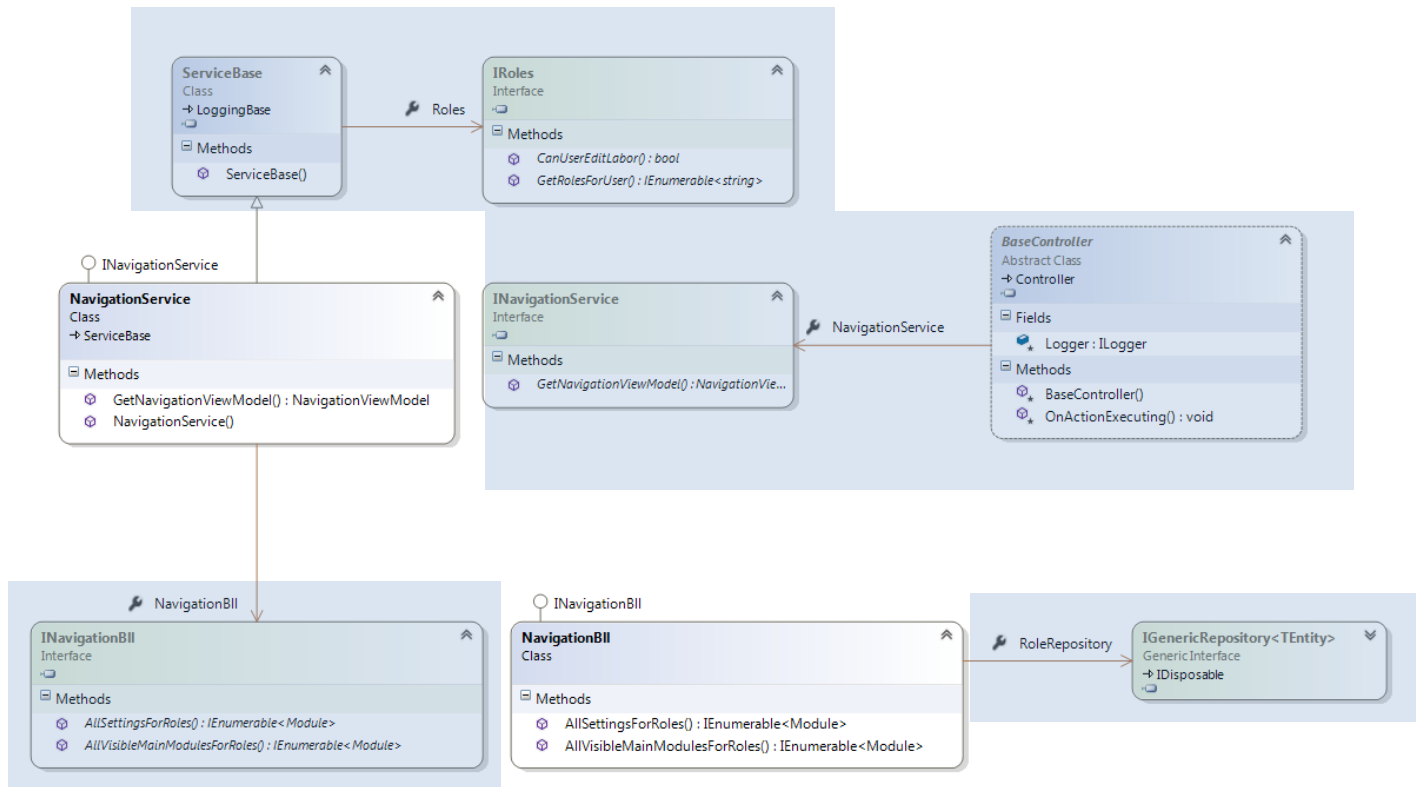
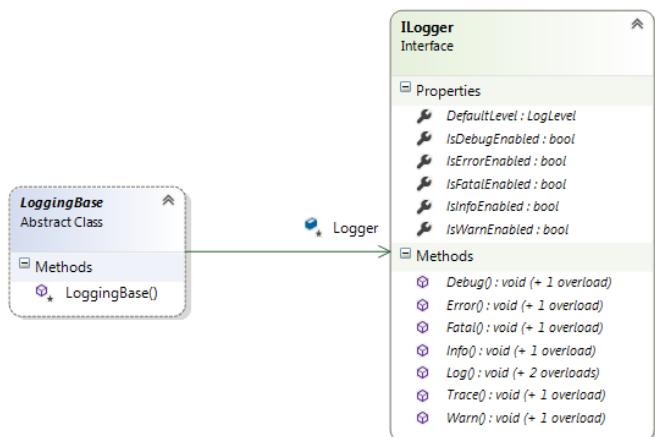


Abbildung 24: Klassendiagramm Navigation

4.6 Logging

Damit man im ganzen Projekt simpel Loggen kann und das Logging Framework auch einfach austauschbar ist, haben wir das Loggen entkoppelt.



Wir haben eine abstrakte Baseklasse mit dem Namen LoggingBase, welcher ein Property ILogger besitzt und dieses im Konstruktor setzt.

Abbildung 25: Klassendiagramm LoggingBase

Service Base, beschrieben in 4.5.3, erbt von Logging Base und initialisiert eine Instanz von ServiceBase im Konstruktor.

```

public ServiceBase( ILogger logger ) : base( logger )
{
    logger.ThrowIfNull( nameof( logger ) );
}

```

Ein Service, nehmen wir gerade den im vorherigen Kapitel beschriebenen NavigationService, hat somit die Logging Funktionalität und gibt im Konstruktor als base Attribut eine Implementation von ILogger mit.

```
public NavigationService( ILoggerFactory loggerFactory ): base( loggerFactory.CreateLogger(
typeof(NavigationService) ) )
{
}
```

Damit nicht jede Klasse, welche Loggen will, einen Logger erstellen muss, haben wir unsere LoggerFactory.

Die LoggerFactory erstellt einen Logger mit dem Namen welcher als Attribut mitgegeben wurde.

Unsere NLoggerFactory erstellt uns, wie der Name schon sagt, ein NLogLogger. Falls man einen anderen Logger benutzen möchte, kann man einfach eine neue LoggerFactory erstellen, welche ILoggerFactory implementiert und im IOC, die neue LoggerFactory als Implementation von ILoggerFactory registrieren, siehe Kapitel IOC 0, und der neue Logger wird benutzt. Damit auch die LogLevel nicht angepasst werden müssen, existiert eine NLogExtensions Klasse, welche NLog.LogLevel zum unserem Enum LogLevel konvertiert.

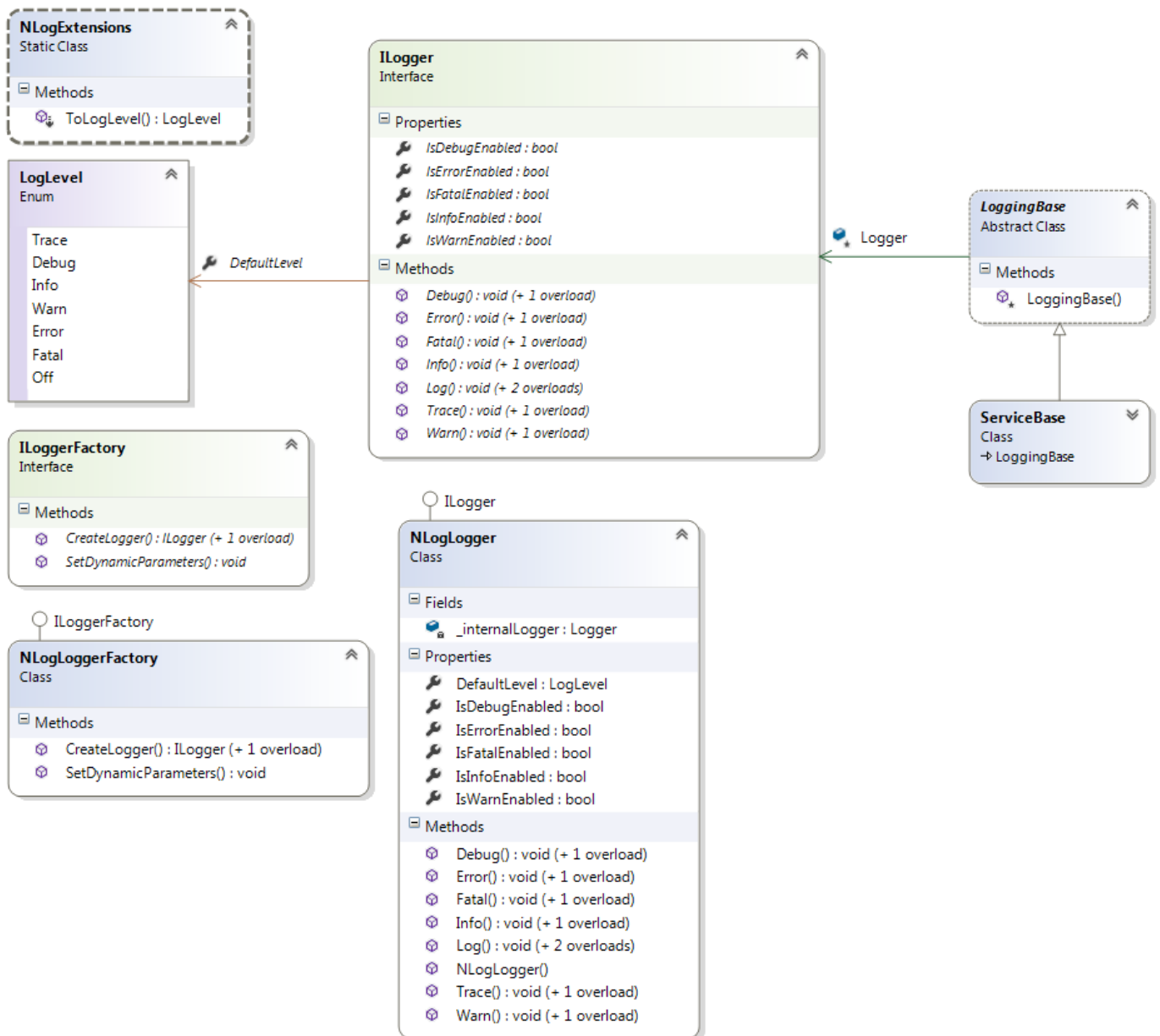


Abbildung 26: Klassendiagramm Logging

Das Diagramm hilft zur Veranschaulichung des erklärten.

4.7 Projekt Diagramm

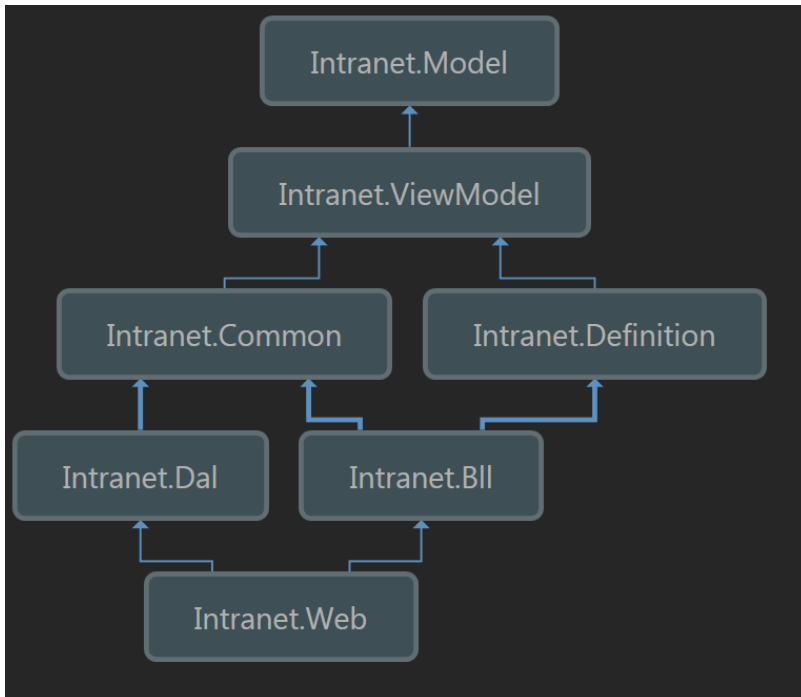


Abbildung 27: Dependencydiagram Shell

Das Abhängigkeitsdiagramm zeigt die Abhängigkeiten zwischen den Projekten der Shell an.⁹

Das Web hat eigentlich nur eine Abhängigkeit auf **Intranet.Bll**, doch dadurch, dass wir im Web mit IOC den **IntranetContext** und die Repositories registrieren besteht auch eine Abhängigkeit auf **Intranet.Dal**. Der Data Access Layer hat eine Abhängigkeit auf **Common**, welches den Datenbankzugriff vereinfacht. Der Business Layer hat eine Abhängigkeit auf **Definition** welche Ihm Interfaces zu Verfügung stellt. Zum Beispiel könnte ein Interface vom Dal implementiert sein und dem Bll somit Daten liefern. Das Definition Projekt hat Abhängigkeit auf das **ViewModel** welches wiederum Abhängigkeit auf das **Model** hat, um die Schnittstelle zu definieren.

4.8 User Design

Das Design für die Shell Applikation wurde farblich am Design der Intigena Webseite¹⁰ angepasst. Da eine Anforderung an die Shell war, dass sie einfach und verständlich gestaltet werden sollte, damit auch unerfahrene User die Shell benutzen können, haben wir versucht, mit möglichst wenig Text und ohne „unbrauchbaren“ Funktionen auszukommen. Was beispielsweise ersichtlich ist an der Startseite der Intranet-Applikation, bei der es, ausser der Navigation am oberen Seitenrand, nur für jedes implementierte Modul einen grossen Button gibt mit einer kurzen Beschreibung.

⁹ Das Dependency Diagramm über die gesamte Applikation befindet sich unter 10.3

¹⁰ Siehe *Abbildung 28: Intigena Webseite*



Abbildung 28: Intigena Webseite www.intigena.ch

Durch die Verwendung von Bootstrap ist die Applikation gemäss Anforderung auf Desktop Computern sowie auf Tablets bedienbar. Zusätzlich, was keine Anforderung war, kann die Applikation auch auf Smartphones brauchbar aufgerufen werden (Siehe 4.8.1.4 Beispiel Mobile).

4.8.1 Mockups und Endergebnis

Die Mockups für die Shell wurden mit dem Tool *Balsamiq Mockups*¹¹ erstellt.

4.8.1.1 Navigation

Die Navigation befindet sich immer fix oben an der Seite. Das erste Element in der Navigationsleiste ist das für die Benutzer wiedererkennbare Firmenlogo, welches zur Startseite der Intranet Applikation führt. Auf das Logo folgen Links zu allen sichtbaren Modulen für welche der jeweilige Benutzer Zugriffsrechte besitzt. Am rechten Ende der Leiste befinden sich Tools für den Administrator wie z.B. Einstellungen. Beim Mockup Design waren diese zuvor noch am Ende der Modulliste angehängt, wir haben uns jedoch dazu im Verlauf der Entwicklung dazu entschieden diese ans rechte Ende zu verschieben um diese für den Administrator besser von den eigentlichen Modulen abzugrenzen.

Im Tablet- und Smartphone-Modus verschwinden die Module und Administratortools in einem Hamburger Menü.



Abbildung 29: Mockup der Navigationsleiste

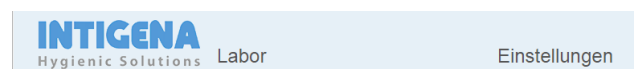


Abbildung 30: Endergebnis der Navigationsleiste

4.8.1.2 Startseite

Die Startseite der Intranet Applikation beinhaltet nur grosse Buttons, je einen für jedes Modul, welcher aus Modulnamen und einer kurzen Beschreibung besteht. Jedoch sieht nicht jeder Benutzer alle Module. Ein Benutzer sieht nur diejenigen Module, für die er auch die jeweilige Berechtigung besitzt und die auch nicht temporär von einem Admin deaktiviert wurden. Die Startseite zeigt einem Benutzer nur das an, was er auch sehen möchte. Auf zusätzliche unnötige Informationen haben wir verzichtet.

¹¹ Siehe: <https://balsamiq.com/products/mockups/>

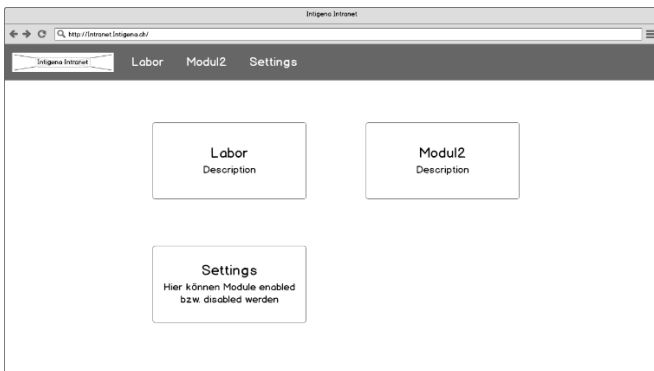


Abbildung 31: Mockup der Intranet Startseite



Abbildung 32: Endergebnis der Intranet Startseite

4.8.1.3 Einstellungen

Die Einstellungsseite der Intranet Applikation ist nur für den Administrator sichtbar und benutzbar. Hier werden einem Administrator alle Module aufgelistet mit Modulnamen und dessen aktuellen Sichtbarkeitsstatus. Durch einen Switch/Toggle kann ein Modul aktiviert bzw. deaktiviert werden. Der Switch im End-design stammt von Bootstrap.

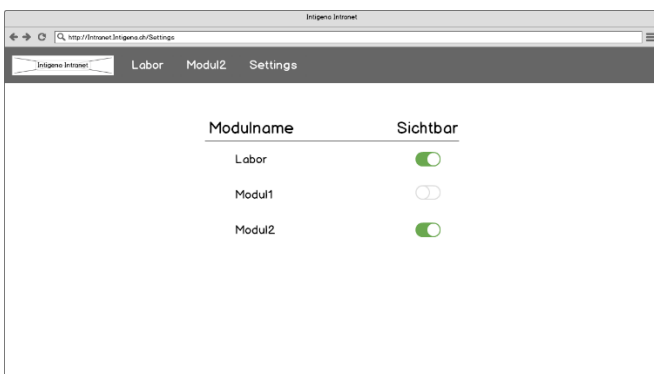


Abbildung 33: Mockup der Einstellungsseite

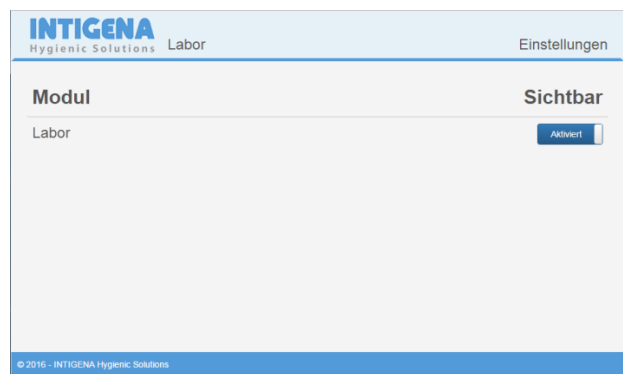


Abbildung 34: Endergebnis der Einstellungsseite

4.8.1.4 Beispiel Mobile

Damit die Intranet-Applikation auch auf kleineren Bildschirmen korrekt und brauchbar benutzt werden kann, verwenden wir fürs Design das freie CSS-Framework Bootstrap. In *Abbildung 35* wird die Startseite auf einem Samsung Galaxy S5 angezeigt. Die Module in der Navigationsleiste verschwinden in einem Hamburger Menü, die grossen Modul-Buttons werden nicht mehr in einem zweier Grid angezeigt, sondern erscheinen untereinander.

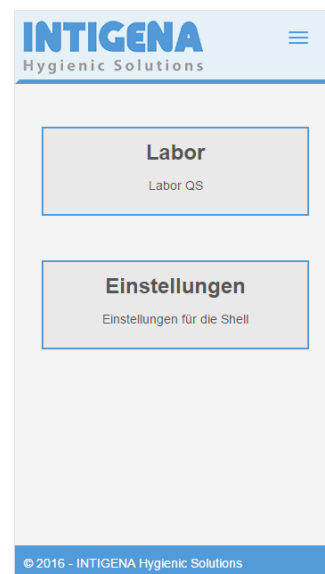


Abbildung 35: Endergebnis der Startseite angezeigt auf einem Samsung S5

5 Labor

5.1 Einleitung

Das Labor/QS Modul dient einerseits zur Erfassung der Tests, welche im Labor durchgeführt werden, andererseits auch für die Darstellung der Testresultate. Ein Benutzer kann auf einem Blick sehen, ob mit der Produktion der Windeln in den letzten 24 Stunden an Hand der eingegebenen Labor-Resultaten noch alles in Ordnung ist. Zusätzlich kann auch auf längst vergangene Testreihen zurückgegriffen werden, falls Kunden mit Reklamationen sich an Intigena wenden.

5.2 Datenbank

Wir haben für das Labor drei verschiedene Datenbanken, welche wir erstellen müssten.

Labor	Die Datenbank bei welcher die Werte welche im Labor erfasst werden abgespeichert werden.
Intigena	Die Stammdatenbank für Intigena für das Labor, welche uns zum Beispiel Daten für die Berechnung der Testresultate liefert.
Instron	Die Datenbank für Instron, welche die Werte von Instron beinhalten könnte. Diese Datenbank wird erstellt, jedoch in unserem Projekt noch nicht verwendet. Dies wurde so geplant.

5.2.1 Labor

Die Datenbank für das Labor soll alle Daten, welche von den Usern erfasst werden, abspeichern können. Auch wurde vorgegeben, dass die Datenbank alle Daten welche berechnet werden abspeichern muss. Somit muss es ein Testblatt abspeichern können, welches entweder ein Testblatt für Babywindeln oder Inko Binden ist. Diese Testblätter besitzen Daten welche für das ganze Blatt gültig ist und verschiedene Tests welche durchgeführt werden in verschiedenen Kategorien. Auch muss man zu den einzelnen Tests Notizen hinterlegen können und einen Fehler auswählen können. Zu den Test Kategorien müssen Mittelwerte und Standardabweichung angezeigt werden. Auch sollen die eingegebenen Daten farbig markiert werden ob sie im akzeptierten Bereich liegen oder nicht.

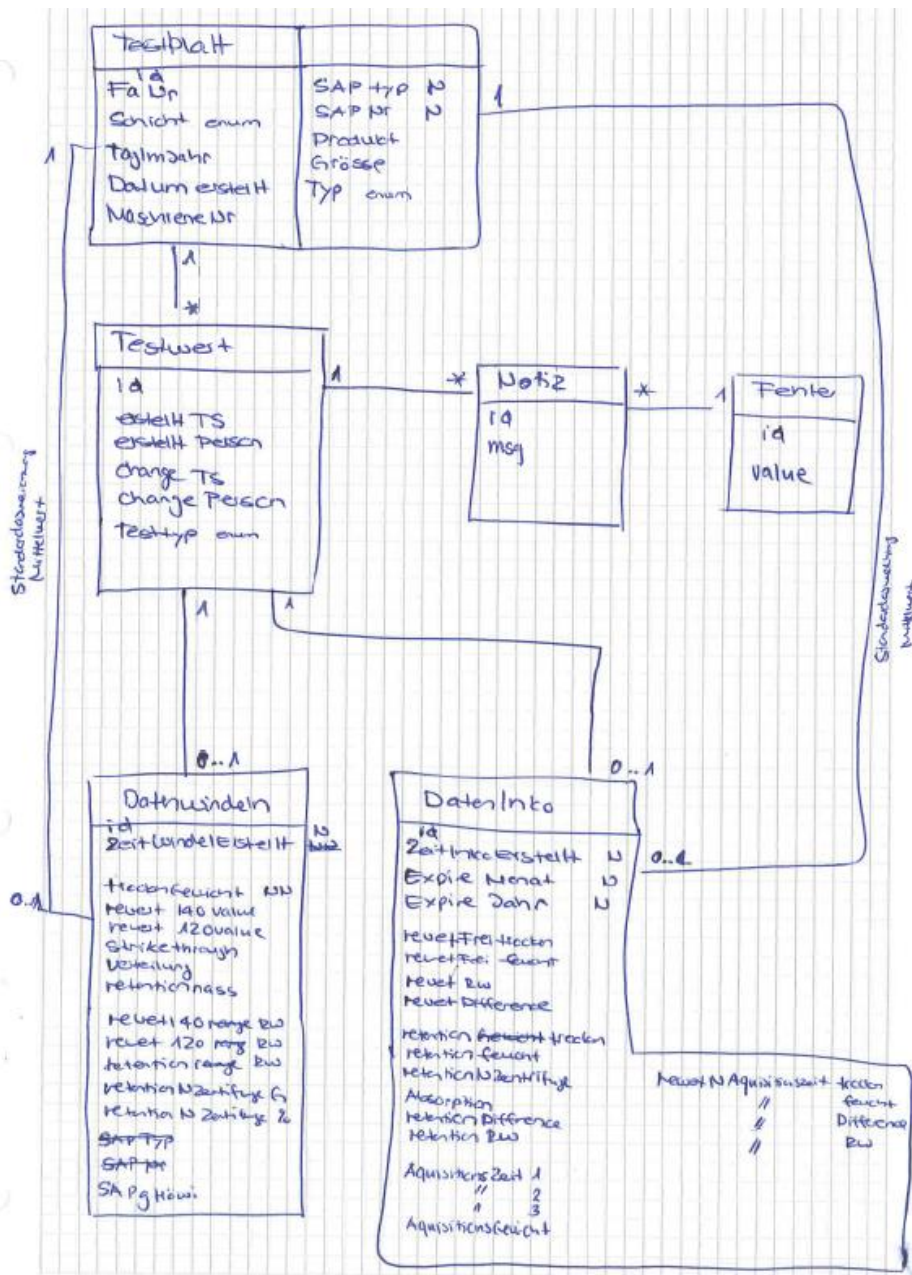


Abbildung 36: Skizze Labor Datenbank

Diese Abbildung zeigt eine Skizze für die Labor Datenbank und wurde während der Datenbankbesprechung erstellt, welche grob unsere Designentscheidungen aufzeigt.

5.2.1.1 Datenbankdiagramm

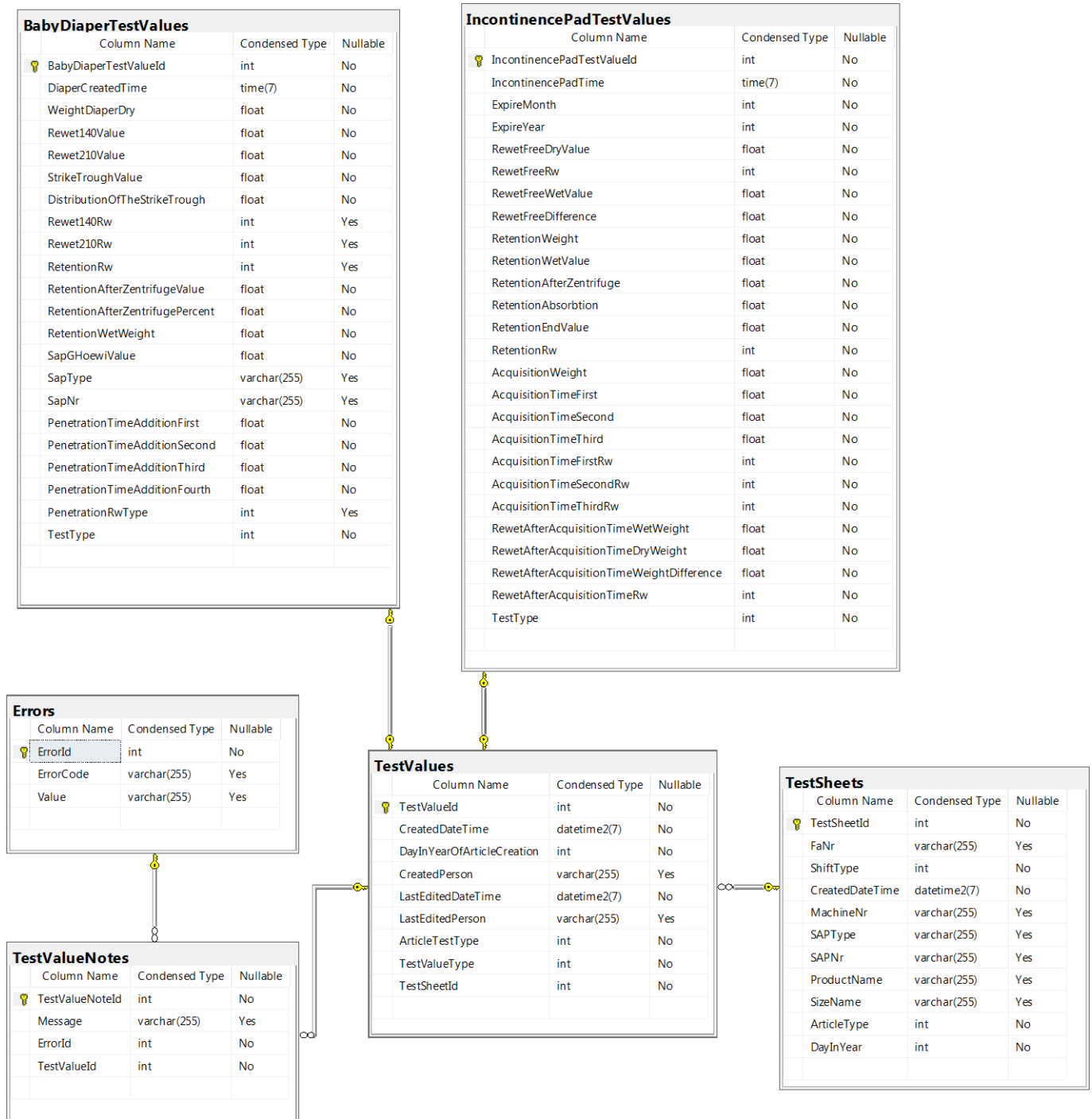


Abbildung 37: Datenbankdiagramm Labor

Dieses Datenbankdiagramm ist das ERM der Finalen Datenbank für das Labor.

Ein TestSheet beinhaltet alle Werte welche zu einem Testblatt bei der ersten Erstellung erfasst werden können, diese Daten kommen unter anderem aus der Datenbank welche wir Intigena benannt haben, siehe Kapitel 5.2.2.

Ein TestSheet beinhaltet beliebig viele TestValues. Ein TestValue kann entweder ein BabyDiaperTestValues oder ein IncontinencePadTestValues sein, welches es ist wird im ArticleTestType hinterlegt wird. Was für einen Test es genau ist, ist im TestTyp hinterlegt.

Ob das TestValue ein einzelner Testwert, ein Durchschnittswert oder eine Standardabweichung ist wird im TestValueType hinterlegt.

Ein TestValues kann verschiedene TestValueNotes beinhalten, welche Notizen von den Benutzern sind. Eine Notiz gehört immer zu genau einem Error. Ein Error kann zu beliebig vielen TestValueNotes gehören.

Eine genaue Beschreibung zu allen Felder befindet sich unter der Url:

<https://claudiasaxer.github.io/Intranet/api/Intranet.Labor.Model.html>

Das Klassendiagramm zum EF Model zum ERM befindet sich im Kapitel 10.5.9.2.

5.2.2 Intigena

In der Stammdatenbank für das Labor hat es eine Fehlertabelle welche Werte und Beschreibungen zu möglichen Fehlern beinhaltet. Diese wird für die Notizen gebraucht. Testblatt Informationen werden anhand der FA Nummer generiert. Zur FA Nummer weiss man dann die Dauer des Auftrages, den Artikel welcher zum Auftrag gehört (zum Beispiel Babywindel oder Inko Binde) und Grenzwerte sowie SAP Werte welche zum Fertigungsauftrag definiert wurden. Auch wird man wissen welche Maschine den Fertigungsauftrag Artikel erstellt.

In der Intigena existiert ein Schichtplan mit Start Tag, Start Ende, Start Zeit, End Zeit und einem Name mit welchem die Mitarbeiter die Schicht kennen. Mit diesen Daten soll automatisch die Schichtdaten für das Testblatt generiert werden.

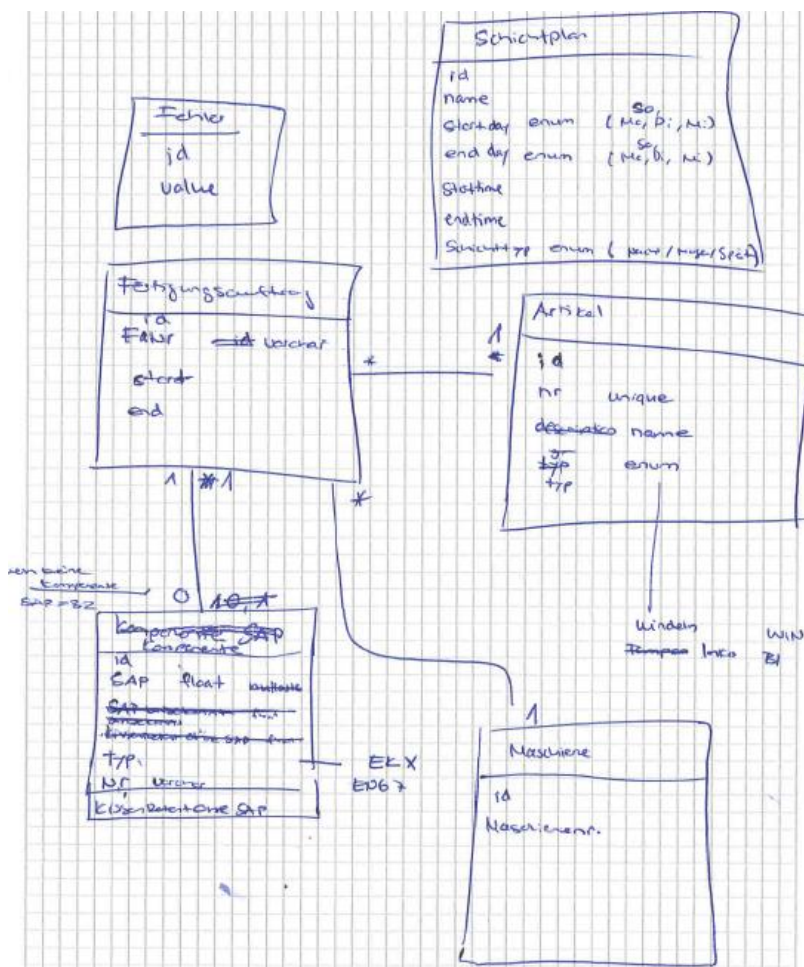


Abbildung 38: Skizze Intigena Stammdaten Datenbank

5.2.2.1 Datenbankdiagramm

5.2.2.1.1 Schichtplan

Der Schichtplan ist ein vordefinierter Plan welcher so bei Intigena geführt wird.

ShiftSchedules			
Column Name	Condensed Type	Nullable	
 ShiftScheduleId	int	No	
Name	varchar(255)	Yes	
StartDay	int	No	
EndDay	int	No	
StartTime	time(7)	No	
EndTime	time(7)	No	
ShiftType	int	No	

Abbildung 39: Datenbankdiagramm Intigena Schicht

	ShiftScheduleID	Name	StartDay	EndDay	StartTime	EndTime	ShiftType
1	1	Mo Nacht	0	1	22:00:00.0000000	04:59:59.0000000	0
2	2	Mo Morgen	1	1	05:00:00.0000000	13:59:59.0000000	1
3	3	Mo Spät	1	1	14:00:00.0000000	22:59:59.0000000	2
4	4	Di Nacht	1	2	22:00:00.0000000	04:59:59.0000000	0
5	5	Di Morgen	2	2	05:00:00.0000000	13:59:59.0000000	1
6	6	Di Spät	2	2	14:00:00.0000000	22:59:59.0000000	2
7	7	Mi Nacht	2	3	22:00:00.0000000	04:59:59.0000000	0
8	8	Mi Morgen	3	3	05:00:00.0000000	13:59:59.0000000	1
9	9	Mi Spät	3	3	14:00:00.0000000	22:59:59.0000000	2
10	10	Do Nacht	3	4	22:00:00.0000000	04:59:59.0000000	0
11	11	Do Morgen	4	4	05:00:00.0000000	13:59:59.0000000	1
12	12	Do Spät	4	4	14:00:00.0000000	22:59:59.0000000	2
13	13	Fr Nacht	4	5	22:00:00.0000000	04:59:59.0000000	0
14	14	Fr Morgen	5	5	05:00:00.0000000	13:59:59.0000000	1
15	15	Fr Spät	5	5	14:00:00.0000000	22:59:59.0000000	2
16	16	Sa Nacht	5	6	22:00:00.0000000	04:59:59.0000000	0

Abbildung 40: Stammdaten Schicht

Name	Der Name unter welchem die User die Schicht kennen.
StartDay	Die Nummer des Tages in der Woche unter welchem die Schicht beginnt.
EndDay	Die Nummer des Tages in der Woche unter welchem die Schicht endet.
StartTime	Die Zeit des Beginnes der Schicht.
EndTime	Die Zeit des Endes der Schicht.
ShiftType	Der Typ der Schicht als Enum (Spät, Früh, Morgen)

Tabelle 4: Schicht DB Werte Beschreibung

5.2.2.1.2 Stammdaten zum Fertigungsauftrag

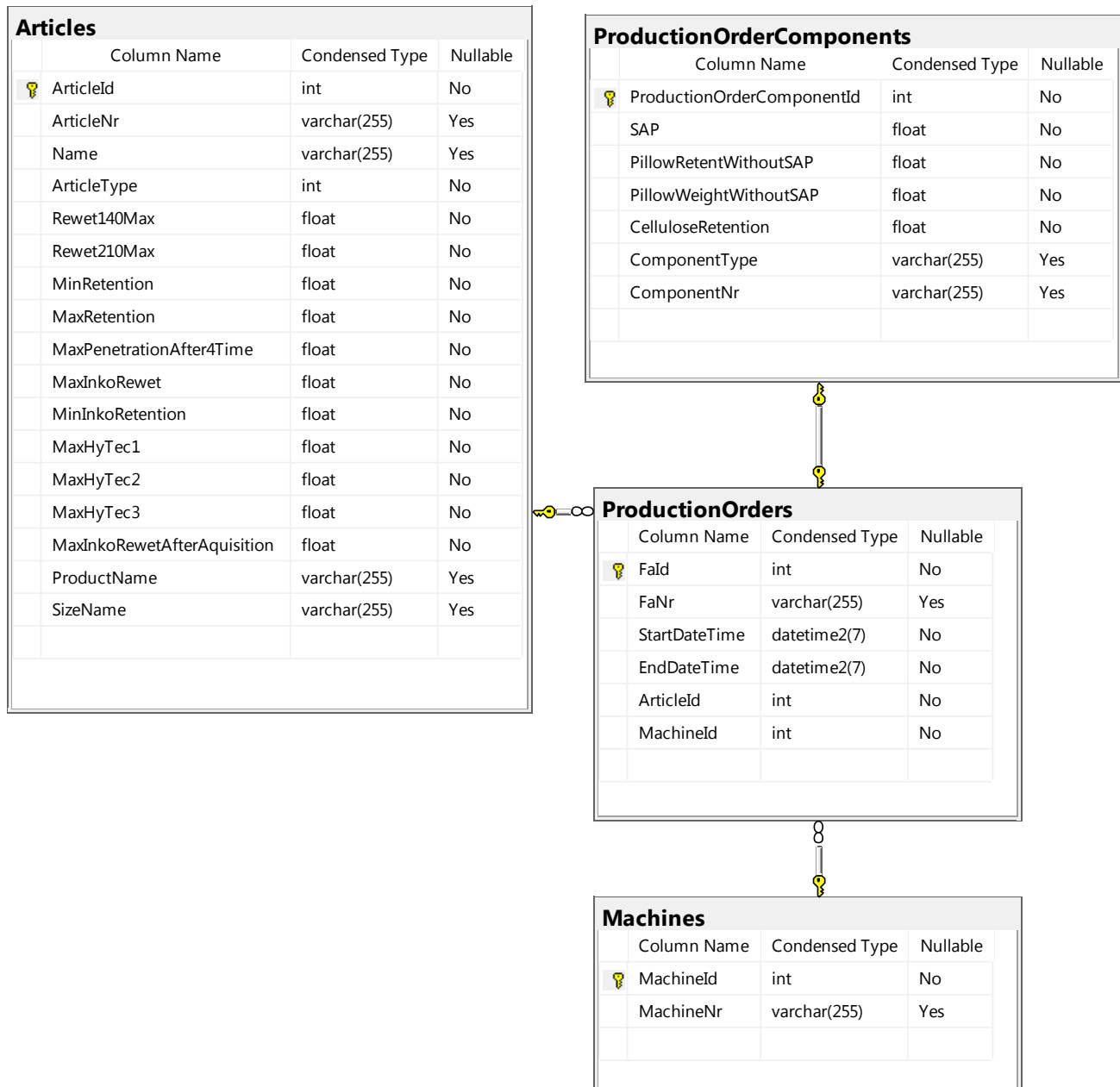


Abbildung 41: Datenbankdiagramm Intigena Stammdaten

Maschine

Die Maschinen werden alle mit ihrer Nummer im System erfasst.

	MachineId	MachineNr
1	1	M11
2	2	M49
3	3	M10

Abbildung 42: Maschine Datenbeispiel

MachineId

Die eindeutige Id für die Maschine

MachineNr

Die Bezeichnung der Maschine. Geläufig als Maschinenummer.

Tabelle 5: Maschine DB Werte Beschreibung

Artikel

	ArticleId	ArticleNr	Name	ArticleType	Rewet140Max	Rewet210Max	MinRetention	MaxRetention	MaxPenetrationAfter4Time
1	1	10401	Babydream Maxi-Plus	0	0.4	0.5	350	380	250
2	2	10412	Inko Extra	1	0	0	0	0	0
3	3	10501	Cresta Extra Inko Extra	1	0	0	0	0	0
	MaxInkoRewet	MinInkoRetention	MaxHyTec1	MaxHyTec2	MaxHyTec3	MaxInkoRewetAfterAquisition	ProductName	SizeName	
	0	0	0	0	0	0	NULL	NULL	
	0	0	0	0	0	0	NULL	Inko Extra	
	0.5	180	20	60	85	2	NULL	NULL	

Abbildung 43: Artikel Datenbeispiel

ArticleId	Die eindeutige Id der Artikels.
ArticleNr	Die Nummer des Artikels.
Name	Der Name unter welchen man den Artikel kennt.
ArticleType	Der Type des Artikels, welcher ein Enum ist. 0 = Baby Windel 1 = Inko Binde
Rewet140Max	Der maximale Wert für den Rewet Test mit 140ml. Wird der Wert überschritten, soll der Wert rot gekennzeichnet werden.
Rewet210Max	Der maximale Wert für den Rewet Test mit 210ml. Wird der Wert überschritten, soll der Wert rot gekennzeichnet werden.
MinRetention	Der minimum Wert für den Retention Test. Wird der Wert unterschritten, soll der Wert rot gekennzeichnet werden.
MaxPenetrationAfter4Time	Der maximale Wert für den 4ten Penetration Test. Wird der Wert überschritten, soll der Wert rot gekennzeichnet werden.
MaxInkoRewet	Der maximale Wert für den Rewet Test der Inko Binden. Wird der Wert überschritten, soll der Wert rot gekennzeichnet werden.
MinInkoRewet	Der minimale Wert für den Rewet Test der Inko Binden. Wird der Wert unterschritten, soll der Wert rot gekennzeichnet werden.
MaxHyTec1	Der Maximale Wert für den ersten HyTec Wert. Wird der Wert überschritten, soll der Wert rot gekennzeichnet werden.
MaxHyTec2	Der Maximale Wert für den zweiten HyTec Wert. Wird der Wert überschritten, soll der Wert rot gekennzeichnet werden.
MaxHyTec3	Der Maximale Wert für den dritten HyTec Wert. Wird der Wert überschritten, soll der Wert rot gekennzeichnet werden.
MaxInkoRewetAfterAquisition	Der Maximale Wert für den Rewet Test nach dem Aquisition Test für die Inko Binden. Wird der Wert überschritten, soll der Wert rot gekennzeichnet werden.
ProductName	Der Produkt Name des Artikels.
SizeName	Der Name welche die Grösse des Artikels beschreibt.

ProductionOrderComponent

Dies sind verschiedene Werte welche pro Production Order (Fertigungsartikel), gesetzt werden, da sie für die Berechnungen der Testwerte benötigt werden.

	ProductionOrderComponentId	SAP	PillowRetentWithoutSAP	PillowWeightWithoutSAP	CelluloseRetention	ComponentType	ComponentNr
1	1	32.7	31.2	26	1.2	EKX	EN67

Abbildung 44: Fertigungsartikelkomponente Datenbeispiel

Fertigungsartikel

Dies ist eine Tabelle für alle Fertigungsartikel welche erfasst wurden. Die Fertigungsartikel haben eine gewisse Lebenszeit, nur dann können Testdaten zu den Fertigungsartikeln erstellt werden. Auch haben sie einen Artikel sowie eine Maschine, welche dem Fertigungsartikel zugewiesen ist.

	Fald	FaNr	StartDateTime	EndDateTime	ArticleId	MachineId
1	1	FA123456	2016-01-01 00:00:00.0000000	2018-01-01 00:00:00.0000000	1	1
2	2	FA666	2016-02-02 00:00:00.0000000	2018-02-02 00:00:00.0000000	2	1
3	3	FA654321	2016-01-01 00:00:00.0000000	2018-01-01 00:00:00.0000000	3	2

Abbildung 45: Fertigungsartikel Datenbeispiel

5.2.3 Instron

Instron ist eine alte Maschine welche Daten zu Zugwerte misst. Zu jenen soll die Fa Nummer sowie die Testperson abgespeichert werden. Sowie die verschiedenen Werte welche Instron erstellt, dazu muss der Name, Mittelwert und die Standardabweichung abgespeichert werden können.

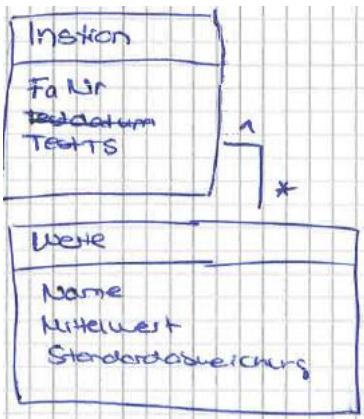


Abbildung 46: Skizze Datenbank Instron

InstronValues			
	Column Name	Condensed Type	Nullable
🔑	InstronValueId	int	No
	Name	varchar(255)	Yes
	Average	float	No
	StandardDevi...	float	No
	IntstronId	int	No



Instrons			
	Column Name	Condensed Type	Nullable
🔑	IntstronId	int	No
	FaNr	varchar(255)	Yes
	TestDateTime	datetime2(7)	No

Abbildung 47: Datenbankdiagramm Instron

Name	Ist der Name des Wertes
Average	ist der Durchschnitt
Standard Deviation	ist die Standardabweichung

Tabelle 6: InstronValues Werte Beschreibung

FaNr	Ist die Fertigungsartikel Nummer für die Instron Werte
TestDateTime	Ist der Timestamp für die Instron Werte

Tabelle 7: Instrons Werte Beschreibung

5.3 Modell

Wir arbeiten mit EF Code First, unser Model widerspiegelt somit die Datenbank. Die Klassendiagramme sind Kapitel 10.5.9 abgelegt.

5.4 Package Diagramm

Ein Package Diagramm über die Shell ist im Kapitel 10.4.210.4.1 abgelegt.

5.5 Projektdiagramm

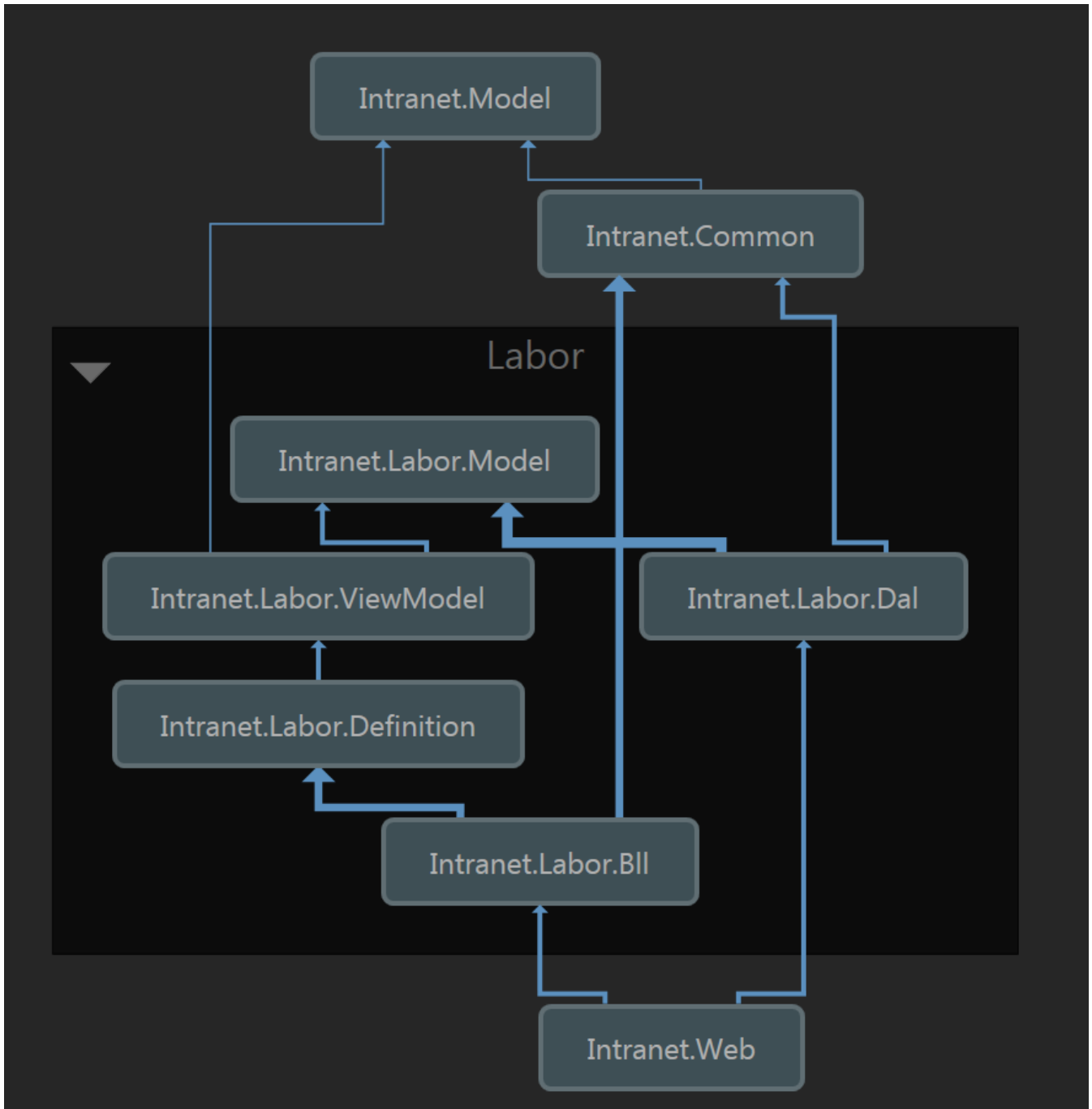


Abbildung 48: Dependency Diagramm Labor

Das Projektdiagramm (Dependency Diagramm) zeigt die Abhängigkeiten zwischen den Projekten des Labors an.¹²

Wie schon bei der Shell hat das Web nur eine Abhängigkeit auf Intranet.Labor.Bll. Wegen der Registrierung fürs IoC gibt's zusätzlich noch eine Abhängigkeit auf Intranet.Labor.Dal. Der Data Access Layer besitzt eine Abhängigkeit auf die Labor Model für die Datenbank und eine aufs Common für die Basisfunktionen. Der Business Layer besitzt Abhängigkeiten aufs Common für die Basisfunktionen und aufs Labor Definition für die Interfaces.

¹² Das Dependency Diagramm über die gesamte Applikation befindet sich unter 10.3

5.6 Sequenzdiagramm

5.6.1 Aufruf eines Testformulars zum Editieren

Fordert ein Client ein Testresultat zum Editieren an, ruft dieser den jeweilig zuständigen Controller mit Edit und der Test ID auf. Der Controller fordert dabei das ViewModel von einem Service an, welcher seinerseits das Model mit der zugehörigen ID von der TestBLL anfordert, welcher das Model von der Datenbank holt. Der Service wandelt das Model ins ViewModel um und schickt dieses wieder an den Controller zurück, welcher seinerseits die View damit abfüllt und an den Client schickt.

Der ganze Vorgang ist noch grafisch in Abbildung 49 dargestellt, welches das Sequenzdiagramm für den Aufruf eines Inko-Rewet Testformulars zeigt.

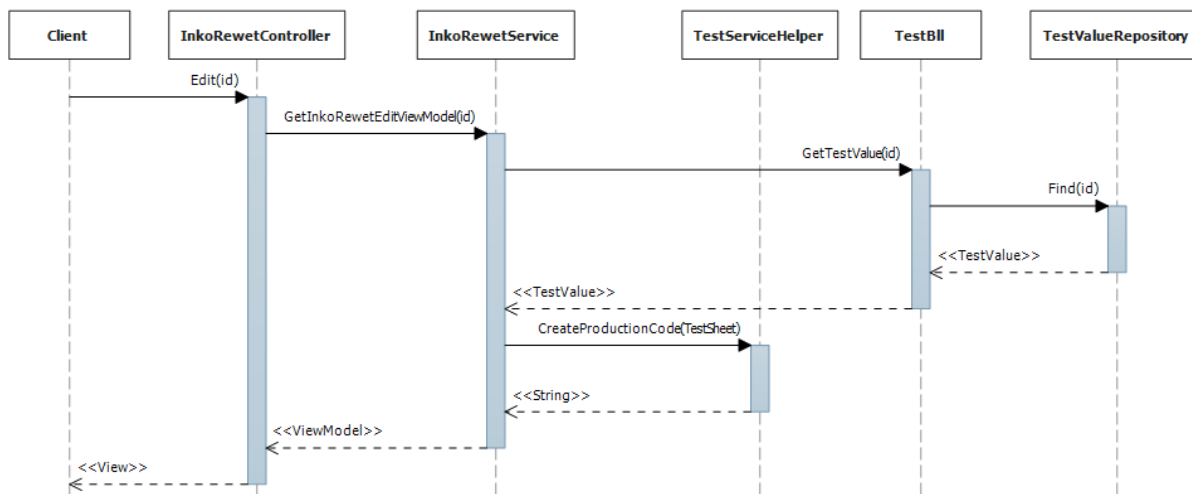


Abbildung 49: Sequenzdiagramm für den Aufruf eines Inko-Rewet Testformulars

5.6.2 Speichern eines Testformulars

Soll ein Testresultat abgespeichert werden, sendet der Client das validierte ViewModel an den Controller weiter, welcher dieses selber weiter an seinen Service leitet. Der Service überprüft, ob das ViewModel ein neues Testresultat ist oder ein altes, welches upgedatet werden soll. Ist es ein neues, so wird mithilfe von ServiceHelpers ein neuer TestValue erstellt, mit den Daten vom ViewModel abgefüllt und weiter am TestBll gegeben, welcher den neuen TestValue in die Datenbank schreibt. Ist es ein altes Testresultat, welches geändert werden soll, so wird das alte Testresultat mithilfe des TestBll von der Datenbank geladen, mit den Daten des ViewModel aktualisiert und wieder zurück in die Datenbank geschrieben.

Wurde der TestValue erfolgreich in der Datenbank abgespeichert, so müssen alle Durchschnitts- und Standardabweichungswerte des dazugehörigen Testblattes (TestSheet) aktualisiert werden. Dazu holt der Service das TestSheet via TestBll von der Datenbank, berechnet und updatet alle Durchschnitts- sowie Standardabweichungswerte und schreibt es wieder Zurück in die Datenbank.

Der ganze Speichervorgang ist noch grafisch in Abbildung 50 dargestellt, welches das Sequenzdiagramm für das Speichern eines Inko-Rewet Testformulars zeigt.

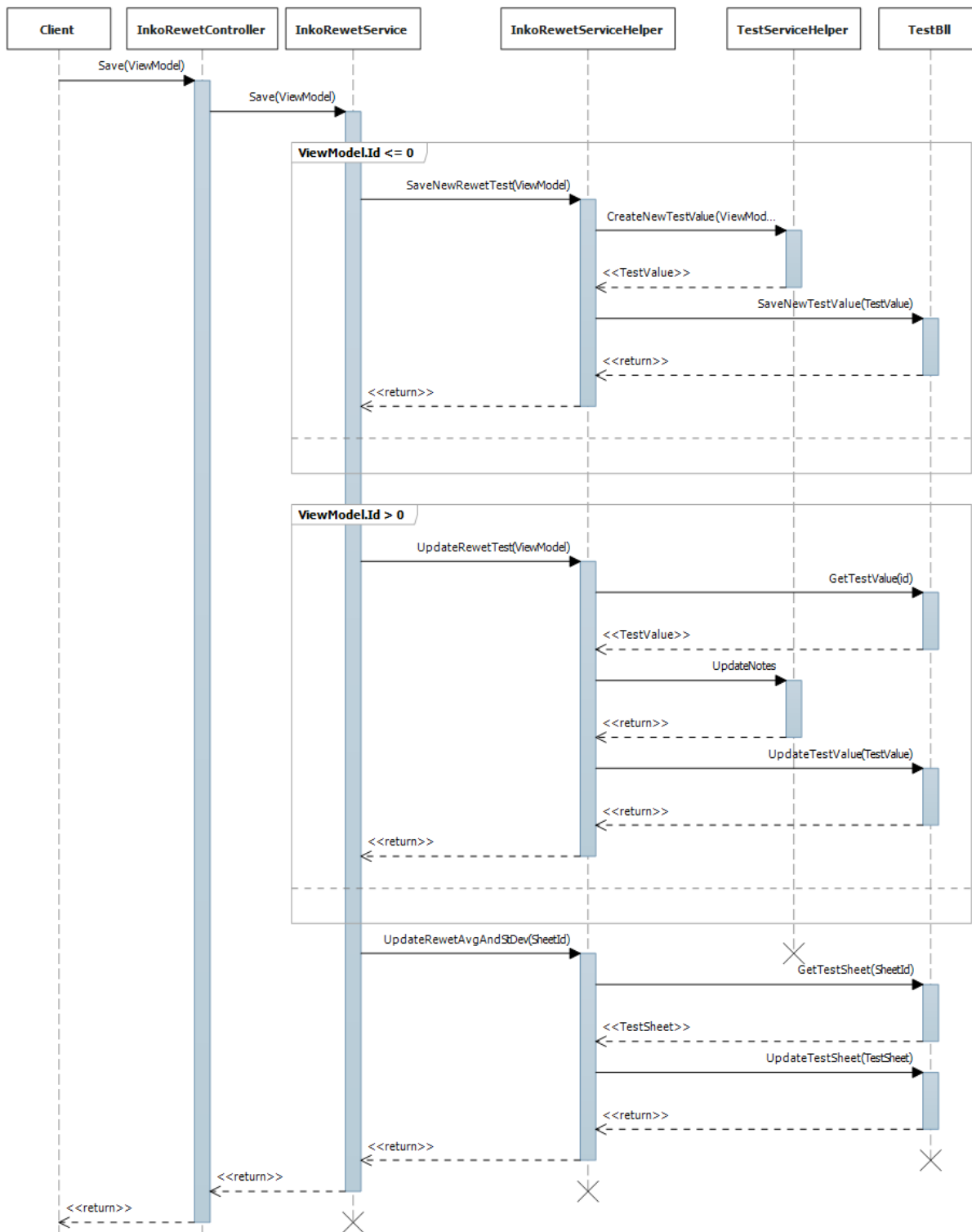


Abbildung 50: Sequenzdiagramm für das Speichern eines Inko-Rewet Testformulars

5.7 User Design

Das Labor-Modul wird in die Shell in einen Body-Container eingebaut. Die Navigation kommt somit von der Shell sowie die Fussleiste.

Die Mockups für das Labor-Modul wurden von Hand gezeichnet.

5.7.1 Startseite

Die Startseite des Labor-Modul ist wie die Startseite¹³ des Intigena Intranets designt, damit das Modul designmässig zum Intranet passt. Sie beinhaltet ebenfalls grosse Buttons mit Namen und je einer kurzen Beschreibung, welche einen Benutzer zu den verschiedenen Unterthemen führt. Deswegen wurde für diese View auf ein Mockup verzichtet. Ein Administrator und ein User mit Schreibrechten sieht die Buttons Testerfassung, Dashboard und History. Bei einem User mit nur Leserechten gibt es den Button Testerfassung nicht.

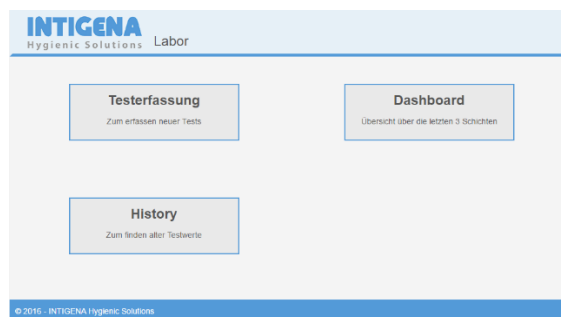


Abbildung 51: Startseite des Labor-Modul

5.7.2 Dashboard

Die Grundanforderung an das Dashboard war, dass auf einem Blick gesehen werden kann, ob mit der Produktion alles in Ordnung ist. Der Kunde wollte sehen, ob in den letzten 24 Stunden (in den letzten drei Schichten) mit einer Maschine etwas nicht stimmte mithilfe eines Ampelsystems.

5.7.2.1 Wireframes

Bei der Entwicklung der Wireframes für das Dashboard orientierten wir uns an einem Vorschlag von unserem Betreuer. Der Vorschlag bestand aus einer Matrix, welche aus vier Spalten (für die letzten drei Schichten plus zusätzlich der aktuellen aktiven Schicht) und 3 Zeilen (für jede Maschine). Das Design dabei wurde ein wenig vom Visual Studio Team Service VSTS Work Board abgeschaut. Die einzelnen Elemente in der Tabelle können dabei aufgeklappt werden um mehr Informationen zu erhalten.

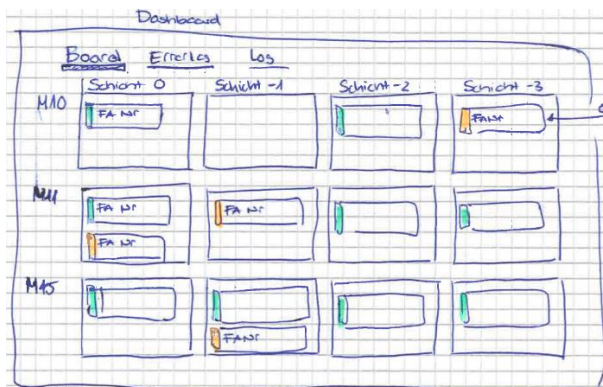


Abbildung 52: Dashboard Wireframe



Abbildung 53: Detailansicht im Dashboard

¹³ Siehe 4.8.1.2

5.7.2.2 Endergebnis

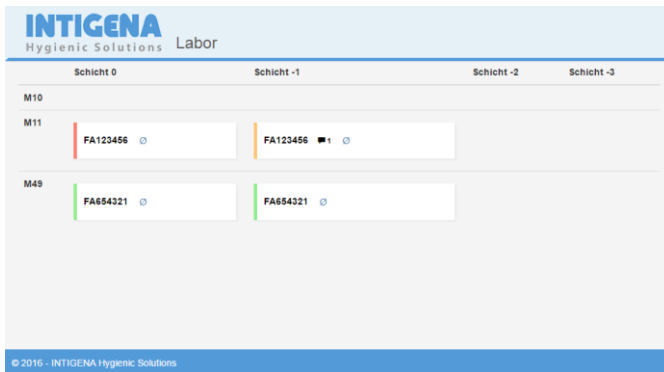


Abbildung 54: Endergebnis des Dashboards

FA123456		
Code	Meldung	Nachricht
080	Fristage fehlt	Testmessage
Inhalt		Wert
Retention - Nach Zentrifuge (g)		360.8
Rewet - 140 ml		0.2
Rewet - 210 ml		0.33
Rewet - 140 ml		0
Rewet - 210 ml		0
Penetrationszeit - Zugabe 4		80

Abbildung 55: Endergebnis der Detailansicht im Dashboard

5.7.3 Testerfassung

Wenn Tests erfasst werden, soll der Benutzer möglichst wenig Eingaben müssen. Die Grundidee fürs Design war, dass es ein einziges Eingabefeld gibt in welches die Fertigungsauftragsnummer eingegeben werden muss, um auf das Testblatt zuzugreifen. Später kam auch noch die Idee dazu, dass bereits existierende Testblätter, welche ein Benutzer in der aktuellen Schicht braucht, unten als Button anzeigen, damit auch ohne Eingabe der FA-Nummer das Testblatt aufgerufen werden kann. Für den Button wurde derjenige aus der Startseite wieder verwendet.

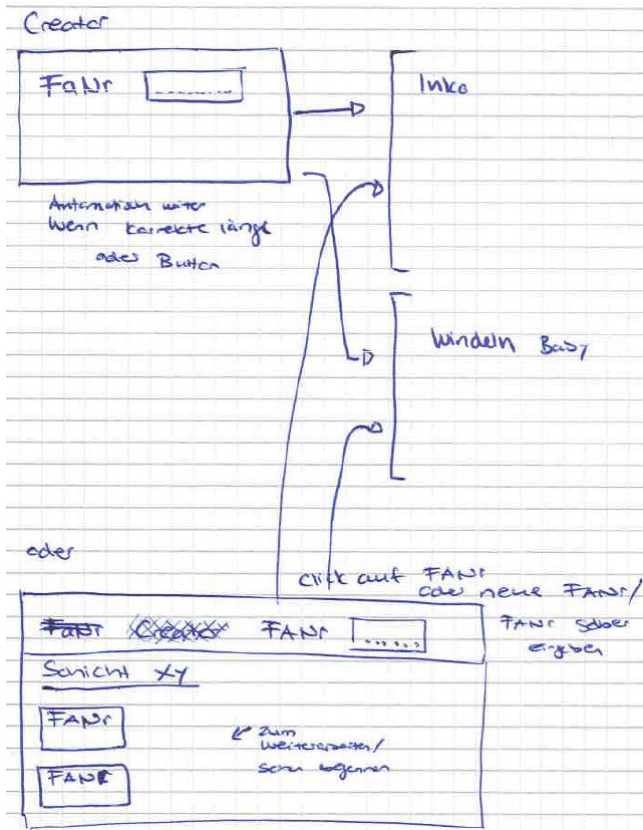
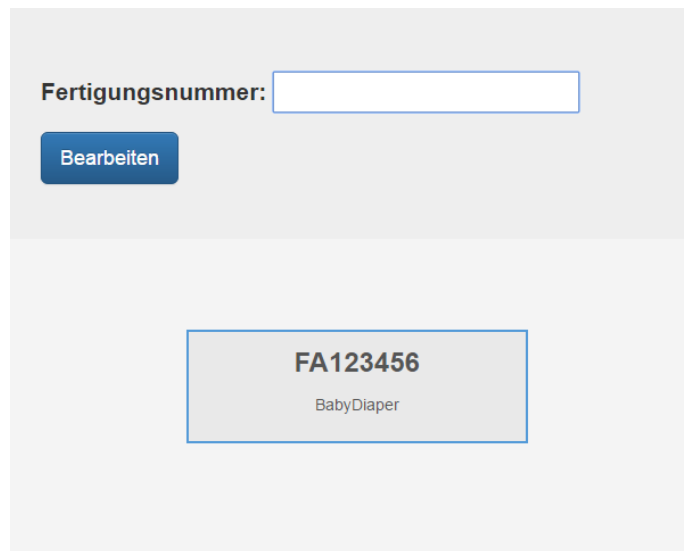


Abbildung 56: Wireframe zur Startseite für die Testerfassung



The screenshot shows the 'Fertigungsnummer:' input field with the value 'FA123456' and the 'BabyDiaper' label below it. The 'Bearbeiten' button is visible next to the input field.

Abbildung 57: Endergebnis der Startseite für die Testerfassung

5.7.3.1 Babywindel

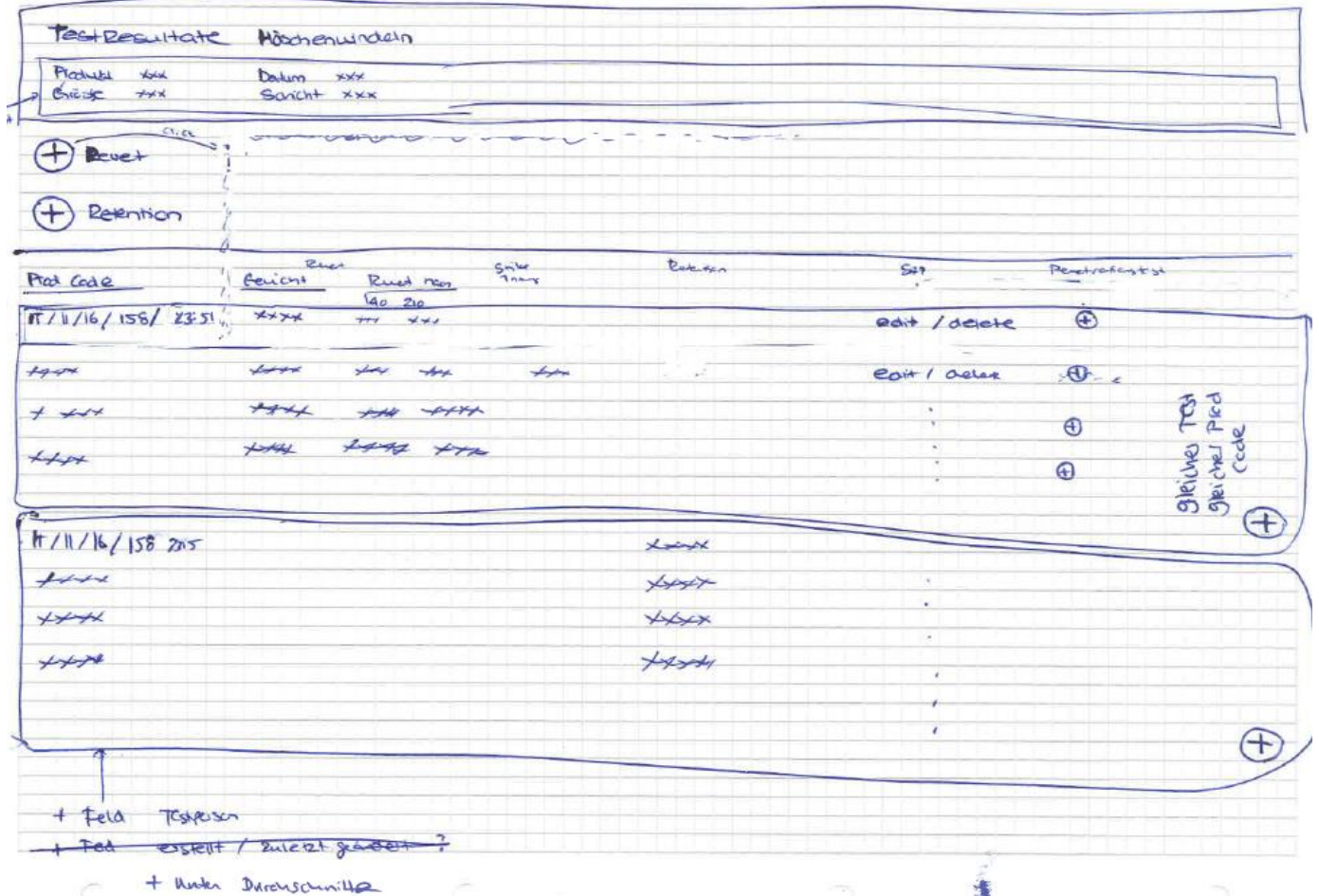
5.7.3.1.1 Detailansicht

Die Idee bei der Detailansicht für das Testblatt der Babywindel war, dass alle Testwerte plus die Durchschnitts- und Standardabweichungs-Werte tabellarisch dargestellt werden wie in der Excel-Sheet Vorlage. Jedoch kamen wir zum Schluss, dass dies Unübersichtlich wäre, da zum ersten es zu viele Spalten geben würde und zum zweiten nicht jede Spalte für jede Testreihe gebraucht wird. Deswegen entschieden wir uns die Testwerte in mehreren thematisierten Tabellen darzustellen.

Wenn einzelne Testwerte nicht in Ordnung sind sollte dies mit einer Farbe signalisiert werden.

Um neue Testwerte hinzuzufügen, gibt es grosse Buttons am Anfang der Seite. Um Testwerte zu bearbeiten oder zu löschen gibt es am Ende einer Testreihe dafür zwei Buttons.

Benutzer ohne Schreibrechte können auch auf die View zugreifen, aber sehen diese Buttons alle nicht.



The wireframe illustrates the layout of the 'Detailansicht' for baby diapers. It is divided into several sections:

- Header Section:** Contains the title 'Test Resultate Windeln' and a sub-header 'Nächstenwindeln'. Below this are fields for 'Produkt xxx', 'Datum xxx', 'Größe xxx', and 'Sort xxx'.
- Action Buttons:** Two large circular buttons labeled '+ Reset' and '+ Retention' are positioned on the left side.
- Main Table:** A table with columns: 'Prod Code', 'Gewicht', 'Rund run', 'Spüle', 'Bekann', 'Sitz', and 'Penetrationstest'. The first row contains placeholder data like '11/11/16/158/2351' and 'xxx'. Subsequent rows show more data with some cells highlighted in red to indicate errors or warnings.
- Footer Section:** Includes a '+ Feld Testreihen' button, a '+ Feld erstellt / zuletzt geändert?' button, and a '+ Noten Durchschnitt' button.
- Annotations:** A note on the right side says 'gleiches Test gleiches Prod Code' with a plus icon. Another note at the bottom right says 'gleiches Test gleiches Prod Code' with a plus icon.

Abbildung 58: Wireframe der Detailansicht für Babywindeln


 Labor

Einstellungen

Testresultate Höschenwindeln - FA123456

Produkt: Babydream Grösse: Maxi-Plus Datum: 28.11.2016 Schicht: Morgen

+ Rewet
+ Retention

Rewet

Prod Code	Gewicht trocken	nach 140 ml	nach 210 ml	Strike Through	Verteilung	Testperson	Edtieren	Löschen
IT/11/16/158/23.51	32.9	0.1	0.18	0.28	240	sj		
IT/11/16/158/23.51	32.1	0.3	0.6	0.27	270	sj		
Standard Abweichung		0.14	0.3	0.01	21.21			
Durchschnitt		0.2	0.39	0.28	255			

Penetrationszeit

Prod Code	Gewicht trocken	Zugabe 1	Zugabe 2	Zugabe 3	Zugabe 4	Testperson	Edtieren	Löschen
IT/11/16/158/23.51	32.9	20	40	60	80	sj		
Standard Abweichung		0	0	0	0			
Durchschnitt		20	40	60	80			

Retention

Prod Code	Gewicht trocken	Gewicht nass	Nach Zentrifuge (g)	Nach Zentrifuge (%)	SAP Typ	SAP Nr.	g/Howl	Testperson	Edtieren	Löschen
IT/11/16/158/23.51	32.2	393	360.8	1120.5	EKX	EN67	10.08	sj		
Standard Abweichung		0	0	0			0			
Durchschnitt		393	360.8	1120.5			10.08			

Allgemeine Werte
 Gewicht trocken - Durchschnitt: 32.4 Gewicht trocken - Standardabweichung: 0.44

© 2016 - INTIGENA Hygienic Solutions

Abbildung 59: Endergebnis der Babywindeln

5.7.3.1.2 Test Erstellung

Das Design für die Erstellung der einzelnen Testreihen ist ein einfaches Formular inkl. Client-Side-Validierung. Es soll möglichst jeden Input vorgenerieren wenn es möglich ist.

Da Penetrationszeit-Tests gleichzeitig mit Rewet Tests gemacht werden, kann man mittels einen Button das Formular für Penetrationszeit Tests mit JavaScript einblenden, bzw. wieder entfernen.

Notizen können auch via Buttonklick hinzu oder wieder entfernt werden.

⊕ Rewet ↓

Testperson:
 Prodcode IT/11/16/158 :
 Gewicht:

Rewet (g) total 3x30 ml
 140ml:
 210ml:
 Strike Through (g) total 3x30 ml
 210 ml:
 mm:

⊕ Penetrationszeit
 ⊕ Notizen

SPEICHERN

Abbildung 60: Wireframe für die Rewet-Testeingabe

Das Formular für Retention-Tests ist vom Aufbau und Design her ähnlich.

Höschenwindeln - Rewet - Test

Testperson:

Produktions Code: IT/11/16/ :

Gewicht [g]:

BabyDiaperRewet (g)
 140 ml:
 210 ml:
 Strike Through:
 Verteilung (mm):

Penetration:

+

Notizen:

+

Speichern

Abbildung 61: Endergebnis des Rewet-Testeingabe-Formulars

5.7.3.2 Inkowindel

Das Design für die Inkowindel-Detailansicht sowie deren einzelnen Testerstellungsformulare ist ähnlich wie die der Babywindel. Deswegen wird hier nicht weiter darauf eingegangen.

5.8 User Ablaufdiagramm

Basierend auf den Ideen der Mockups wurde folgendes Ablaufdiagramm erstellt, welches die Möglichkeiten eines Benutzers in der Anwendung zeigt. Die gelben Kästchen stellen dabei immer eine View dar. Beim Testblatt und den einzelnen Testwert-Formularen wurde hierbei nicht unter den verschiedenen Typen unterschieden, da sich diese mit wenigen Ausnahmen gleich verhalten. Zum schwarzen Initial-Point kann ein Benutzer jederzeit von jedem Ort her hingelangen via der Navigationsbar.

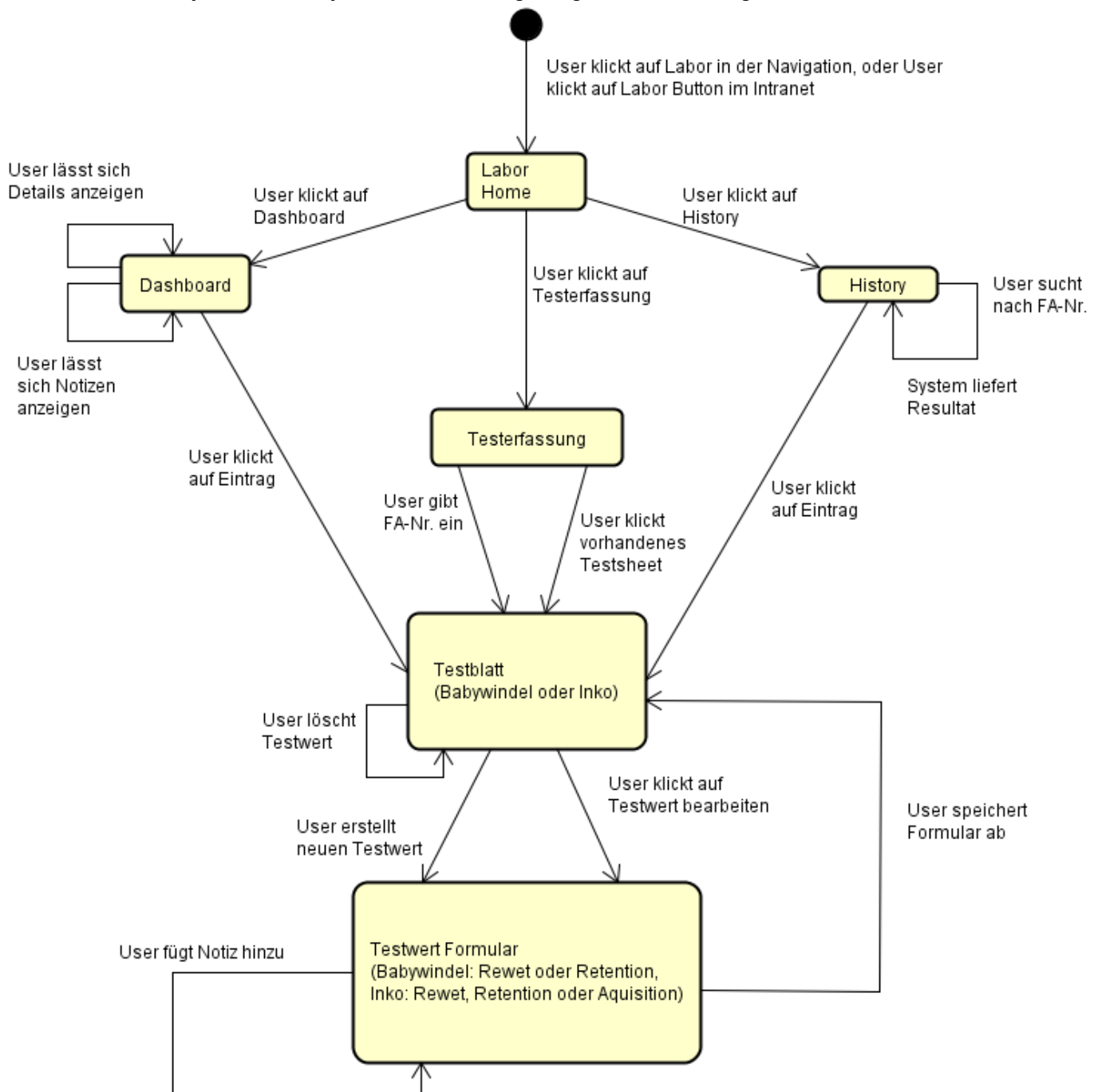


Abbildung 62: Ablaufdiagramm aus der Sicht eines Benutzers

6 Testplan

Der Systemtest wird auf einem Testserver mit vordefinierten Active Directory durchgeführt. Die Testserveradresse lautet: <http://sinv-56013.edu.hsr.ch>

Auf dem Testserver wurde für jede existierende Intranet Rolle ein Testbenutzer erstellt. Die Zugangsdaten für die verschiedenen Benutzer auf dem Testserver sind in der folgenden Liste ersichtlich:

- Administrator-Benutzer:
 - Login: Admin
 - Passwort: Qwertzuiop1!
- Labor-Benutzer mit Lese- und Schreibrechten (LaborUser):
 - Login: LaborTester
 - Passwort: Qwertzuiop1!
- Labor-Benutzer mit nur Leserechten (LaborViewer):
 - Login: LaborReader
 - Passwort: Qwertzuiop1!
- Benutzer ohne Labor Rechte (NotLaborUser):
 - Login: NotLaborUser
 - Passwort: Qwertzuiop1!

6.1 Shell

Test 1	
Funktionalität	Deaktivieren eines Modules
Vorbedingung	• Als Admin eingeloggt • Ein Modul mit Name Labor existiert
Abhängigkeiten	-
Ablauf	1. Intranet-Startseite aufrufen 2. Auf Einstellungen oben rechts in der Navigationsleiste navigieren 3. Slider hinter Modul Labor deaktivieren 4. Zurück zur Startseite navigieren 5. Manuell URL aufrufen ({host}/Labor/LaborHome)
Erwartete Resultate	Bei 4. existiert das Modul Labor nicht mehr (weder in der Navigationsleiste noch als Button) 5. Gibt eine Fehlermeldung zurück

Test 2	
Funktionalität	Aktivieren eines Modules
Vorbedingung	- Als Admin eingeloggt - Ein Modul mit Name Labor existiert - Das Modul Labor ist deaktiviert
Abhängigkeiten	Test 1
Ablauf	- Intranet-Startseite aufrufen - Auf Einstellungen oben rechts in der Navigationsleiste navigieren - Slider hinter Modul Labor aktivieren - Zurück zur Startseite navigieren - Button Labor anklicken - Manuell URL aufrufen ({host}/Labor/LaborHome)
Erwartete Resultate	Navigation auf Labor Home Seite

Test 3	
Funktionalität	Keine Rechte auf Settings als LaborUser
Vorbedingung	Als LaborUser eingeloggt
Abhängigkeiten	-
Ablauf	- Intranet-Startseite aufrufen - Manuell URL aufrufen ({host}/Settings)
Erwartete Resultate	Weder der Button für die Settings existiert noch die Navigation. Der User wird herausgefordert, sich mit einem berechtigten Account einzuloggen.

Test 4	
Funktionalität	Keine Rechte auf Settings als LaborViewer
Vorbedingung	Als LaborViewer eingeloggt
Abhängigkeiten	-
Ablauf	- Intranet-Startseite aufrufen - Manuell URL aufrufen ({host}/Settings)
Erwartete Resultate	Weder der Button für die Settings existiert noch die Navigation. Der User wird herausgefordert, sich mit einem berechtigten Account einzuloggen.

Test 5	
Funktionalität	Keine Rechte auf Labor oder Settings als NotLaborUser
Vorbedingung	Als NotLaborUser eingeloggt
Abhängigkeiten	-
Ablauf	- Intranet-Startseite aufrufen - Manuell URL aufrufen {host}/Settings {host}/Labor
Erwartete Resultate	Weder der Button für die Settings oder das Labor existiert noch die Navigation. Der User wird herausgefordert, sich mit einem berechtigten Account einzuloggen.

Test 6	
Funktionalität	Rechte auf Labor als LaborUser
Vorbedingung	Als LaborUser eingeloggt
Abhängigkeiten	-
Ablauf	- Intranet-Startseite aufrufen - Zu Labor navigieren - Manuell URL aufrufen ({host}/Labor) - Auf Labor Button in der Startseite klicken - Auf Labor Button in der Navigation klicken
Erwartete Resultate	Der Button für das Labor existiert auf der Startseite sowie in der Navigation. Der User wird für 2a und 2b auf die Laborstartseite weitergeleitet.

Test 7	
Funktionalität	Rechte auf Labor als LaborViewer
Vorbedingung	Als LaborUser eingeloggt
Abhängigkeiten	-
Ablauf	- Intranet-Startseite aufrufen - Zu Labor navigieren - Manuell URL aufrufen ({host}/Labor) - Auf Labor Button in der Startseite klicken - Auf Labor Button in der Navigation klicken
Erwartete Resultate	Der Button für das Labor existiert auf der Startseite sowie in der Navigation. Der User wird für 2a und 2b auf die Laborstartseite weitergeleitet.

6.2 Labor

6.2.1 Zugriffsrechte

Test 8	
Funktionalität	Rechte auf LaborCreator als LaborUser
Vorbedingung	- Als LaborUser eingeloggt - Labor-Startseite aufgerufen
Abhängigkeiten	-
Ablauf	1. Auf Testerfassung Button klicken
Erwartete Resultate	Der Button für die Testerfassung existiert auf der Startseite. Der User weitergeleitet zur Testerfassung.

Test 9	
Funktionalität	Rechte auf LaborCreator als Admin
Vorbedingung	- Als Admin eingeloggt - Labor-Startseite aufgerufen
Abhängigkeiten	-
Ablauf	1. Auf Testerfassung Button klicken
Erwartete Resultate	Der Button für die Testerfassung existiert auf der Startseite. Der User weitergeleitet zur Testerfassung.

Test 10	
Funktionalität	Rechte auf LaborCreator als LaborViewer
Vorbedingung	- Als LaborViewer eingeloggt - Labor-Startseite aufgerufen
Abhängigkeiten	-
Ablauf	1. Manuell URL aufrufen ({host}/Labor/LaborCreator)
Erwartete Resultate	Der Button für die Testerfassung existiert nicht auf der Startseite. Der User wird herausgefordert, sich mit einem berechtigten Account einzuloggen.

Test 11	
Funktionalität	Rechte auf LaborDashboard als LaborUser
Vorbedingung	- Als LaborUser eingeloggt - Labor-Startseite aufgerufen
Abhängigkeiten	-
Ablauf	1. Auf Dashboard Button klicken
Erwartete Resultate	Der Button für das Dashboard existiert auf der Startseite. Der User weitergeleitet zum Dashboard.

Test 12	
Funktionalität	Rechte auf LaborDashboard als LaborViewer
Vorbedingung	- Als LaborViewer eingeloggt - Labor-Startseite aufgerufen
Abhängigkeiten	-
Ablauf	1. Auf Dashboard Button klicken
Erwartete Resultate	Der Button für das Dashboard existiert auf der Startseite. Der User weitergeleitet zum Dashboard.

Test 13	
Funktionalität	Rechte auf LaborDashboard als Admin
Vorbedingung	- Als Admin eingeloggt - Labor-Startseite aufgerufen
Abhängigkeiten	-
Ablauf	1. Auf Dashboard Button klicken
Erwartete Resultate	Der Button für das Dashboard existiert auf der Startseite. Der User weitergeleitet zum Dashboard.

Test 14	
Funktionalität	Rechte auf History als LaborUser
Vorbedingung	- Als LaborUser eingeloggt - Labor-Startseite aufgerufen
Abhängigkeiten	-
Ablauf	1. Auf History Button klicken
Erwartete Resultate	Der Button für die History existiert auf der Startseite. Der User weitergeleitet zur History.

Test 15	
Funktionalität	Rechte auf History als LaborViewer
Vorbedingung	- Als LaborViewer eingeloggt - Labor-Startseite aufgerufen
Abhängigkeiten	-
Ablauf	1. Auf History Button klicken
Erwartete Resultate	Der Button für die History existiert auf der Startseite. Der User weitergeleitet zur History.

Test 16	
Funktionalität	Rechte auf History als Admin
Vorbedingung	- Als Admin eingeloggt - Labor-Startseite aufgerufen
Abhängigkeiten	-
Ablauf	1. Auf History Button klicken
Erwartete Resultate	Der Button für die History existiert auf der Startseite. Der User weitergeleitet zur History.

6.2.2 Testerfassung

6.2.2.1 Baby Windeln

Test 17	
Funktionalität	Falsche FA Nummer eingeben
Vorbedingung	Als LaborUser auf der LaborCreator Startseite
Abhängigkeiten	Test 8
Ablauf	1. Falsche FaNummer eingeben „fa“ eingeben „fa23“ eingeben „13357“ eingeben „FA“ eingeben und Bearbeiten drücken „FA5“ eingeben und Bearbeiten drücken - Feld leer lassen und Bearbeiten drücken „asdf“ eingeben und Bearbeiten drücken
Erwartete Resultate	1a, 1b, 1c, Fehlermeldung wird angezeigt: „Fertigungsnummer muss mit FA beginnen und mit Nummern Enden.“ 1d, Fehlermeldung wird angezeigt: „Fertigungsnummer muss mindestens 3 Zeichen lang sein.“ 1e, Fehlermeldung wird angezeigt: „Fertigungsnummer wurde nicht gefunden.“ 1f, Fehlermeldung wird angezeigt: „Die Fertigungsnummer ist zwingend.“ 1g, Fehlermeldung wird angezeigt: „Fertigungsnummer muss mit FA beginnen und mit Nummern Enden.“ und Bearbeiten Button löst nichts aus.

Test 18	
Funktionalität	Baby Windel Testblatt erstellen
Vorbedingung	- Als LaborUser auf der LaborCreator Startseite - FA-Nummer FA123456 existiert und ist ein Baby Windel Artikel - Testblatt zur FA-Nummer wurde in der aktuellen Schicht noch nicht erstellt
Abhängigkeiten	Test 8
Ablauf	- FA-Nummer FA123456 eingeben - Bearbeiten drücken
Erwartete Resultate	Weiterleitung auf Testblatt für Baby Windel mit FA-Nummer FA123456. Alle Werte für die Standardabweichung und den Durchschnitt sind 0. Es existieren keine Testdaten. Die Button Rewet und Retention existieren. Datum ist auf das aktuelle Datum gesetzt. Schicht ist auf die aktuelle Schicht gesetzt. Grösse und Produkt Text wurde von der Datenbank übernommen.

Test 19	
Funktionalität	Baby Windel Testblatt öffnen
Vorbedingung	- Als LaborUser auf der LaborCreator Startseite - FA-Nummer FA123456 existiert und ist ein Baby Windel Artikel - Testblatt zur FA-Nummer wurde in der aktuellen Schicht schon erstellt
Abhängigkeiten	Test 18
Ablauf	- Testblatt öffnen - Button mit der FA-Nummer FA123456 drücken - FA-Nummer FA123456 eingeben und bearbeiten drücken
Erwartete Resultate	Weiterleitung auf Testblatt für Baby Windel mit FA-Nummer FA123456.

Test 20	
Funktionalität	Baby Windel Testblatt bearbeiten. Rewet mit Penetrationszeit Test erstellen.
Vorbedingung	- Als LaborUser auf dem Testblatt mit der Fa-Nummer FA123456 - FA-Nummer FA123456 existiert und ist ein Baby Windel Artikel - Testblatt zur FA-Nummer wurde in der aktuellen Schicht schon erstellt und ist leer
Abhängigkeiten	Test 18
Ablauf	- Rewet Button drücken - Daten für Rewet eingeben - Testperson: ab - Produktions Code: 158 - Zeit: 12:00 - Gewicht: 32.9 140 ml = 0.1 210 ml = 0.1 - Strike Trough: 0.28 - Verteilung: 240 - Penetration Button drücken - Daten für Penetration eingeben - Zugabe 1: 20 - Zugabe 2: 40 - Zugabe 3: 60 - Zugabe 4: 80 - Speichern Button drücken
Erwartete Resultate	Im Rewet existiert nun 1 Eintrag mit den Daten: - Prod Code: IT/11/16/158/12:00 - Gewicht trocken: 32.9 - nach 140 ml: 0.1 - nach 210 ml: 0.1 - Strike Trough: 0.28 - Verteilung: 240 - Testperson: ab Die Rewet Standardabweichung ist immer noch konstant 0. Der Rewet Durchschnitt ist: - Nach 140 ml: 0.1 - Nach 210 ml: 0.1 - Strike Trough: 0.28 - Verteilung: 240 Im Penetrationszeit existiert nun 1 Eintrag mit den Daten: - Prod Code: IT/11/16/158/12:00 - Gewicht trocken: 32.9 - Zugabe 1: 20 - Zugabe 2: 40 - Zugabe 3: 60 - Zugabe 4: 80 - Testperson: ab Die Penetrationszeit Standardabweichung ist immer noch konstant 0. Der Penetrationszeit Durchschnitt ist: - Zugabe 1: 20 - Zugabe 2: 40

	- Zugabe 3: 60 - Zugabe 4: 80 Bei den Allgemeinen Werte sind nun die Werte angepasst - Gewicht trocken - Durchschnitt: 32.9 - Gewicht trocken - Standardabweichung: n. def. Keine Felder wurden markiert.
--	---

Test 21	
Funktionalität	Baby Windel Testblatt bearbeiten. Rewet mit Penetrationszeit Test erstellen. Rote und Orange Markierung sowie Standardabweichung, Durchschnitt und allgemeine Werte testen.
Vorbedingung	• Als LaborUser auf dem Testblatt mit der Fa-Nummer FA123456 • FA-Nummer FA123456 existiert und ist ein Baby Windel Artikel • Testblatt zur FA-Nummer wurde in der aktuellen Schicht schon erstellt und beinhaltet die Daten vom Test20
Abhängigkeiten	Test 20
Ablauf	1. Rewet Button drücken 2. Daten für Rewet eingeben a. Gewicht: 32.1 b. 140 ml = 0.2 c. 210 ml = 1.1 d. Strike Trough: 0.32 e. Verteilung: 260 3. Penetration Button drücken 4. Daten für Penetration eingeben a. Zugabe 1: 40 b. Zugabe 2: 82 c. Zugabe 3: 100 d. Zugabe 4: 260 5. Speichern Button drücken
Erwartete Resultate	Nach 1 sind die Felder Testperson mit ab, Produktions Code mit 158 und Zeit mit 12:00 ausgefüllt. Im Rewet existiert nun 1 zusätzlicher Eintrag mit den Daten: - Prod Code: IT/11/16/158/12:00 - Gewicht trocken: 32.1 - nach 140 ml: 0.2 - nach 210 ml: 1.1 - Strike Trough: 0.32 - Verteilung: 260 - Testperson: ab Die Rewet Standardabweichung ist: - Nach 140 ml: 0.07 - Nach 210 ml: 0.71 - Strike Trough: 0.03 - Verteilung: 14.14 Der Rewet Durchschnitt ist: - Nach 140 ml: 0.15 - Nach 210 ml: 0.6 - Strike Trough: 0.3 - Verteilung: 250

Im Penetrationszeit existiert nun 1 zusätzlicher Eintrag mit den Daten:

- Prod Code: IT/11/16/158/12:00
- Gewicht trocken: 32.9
- Zugabe 1: 40
- Zugabe 2: 82
- Zugabe 3: 100
- Zugabe 4: 260
- Testperson: ab

Die Penetrationszeit Standardabweichung ist:

- Zugabe 1: 14.14
- Zugabe 2: 29.7
- Zugabe 3: 28.28
- Zugabe 4: 127.28

Der Penetrationszeit Durchschnitt ist:

- Zugabe 1: 30
- Zugabe 2: 61
- Zugabe 3: 80
- Zugabe 4: 170

Bei den Allgemeinen Werte sind nun die Werte angepasst

- Gewicht trocken - Durchschnitt: 32.5
- Gewicht trocken - Standardabweichung: 0.57

Feld Rewet für den neuen Eintrag, nach 210 ml wurde rot markiert. Gleiche Spalte Durchschnitt wurde auch rot markiert.

Feld Penetrationszeit für den neuen Eintrag, Zugabe 4 wurde rot markiert. Gleiche Spalte Durchschnitt wurde orange markiert.

Test 22	
Funktionalität	Baby Windel Testblatt bearbeiten. Retention Test erstellen.
Vorbedingung	- Als LaborUser auf dem Testblatt mit der Fa-Nummer FA123456 - FA-Nummer FA123456 existiert und ist ein Baby Windel Artikel - Testblatt zur FA-Nummer wurde in der aktuellen Schicht schon erstellt und Retentiontests sind leer.
Abhängigkeiten	Test 18
Ablauf	- Retention Button drücken - Daten für Retention eingeben - Testperson: ab - Produktions Code: 158 - Zeit: 12:00 - Gewicht: 32.2 - Nass: 393 - Speichern Button drücken
Erwartete Resultate	Im Retention existiert nun 1 Eintrag mit den Daten: - Prod Code: IT/11/16/158/12:00 - Gewicht trocken: 32.2 - Gewicht nass: 393 - Nach Zentrifuge (g): 360.9 - Nach Zentrifuge (%): 1120.5 - SAP Typ: EKX - SAP Nr. EN67 - g/Höwi 10.08 - Testperson: ab Die Retention Standardabweichung ist immer noch konstant 0. Der Retention Durchschnitt ist: - Gewicht nass: 393 - Nach Zentrifuge (g): 360.9 - Nach Zentrifuge (%): 1120.5 - g/Höwi 10.08 Bei den Allgemeinen Werte sind nun die Werte verändert. Keine Felder wurden neu markiert.

Test 23	
Funktionalität	Baby Windel Testblatt bearbeiten. Retention Test erstellen und grüne Markierung sowie Standardabweichung und Durchschnitt testen.
Vorbedingung	- Als LaborUser auf dem Testblatt mit der Fa-Nummer FA123456 - FA-Nummer FA123456 existiert und ist ein Baby Windel Artikel - Testblatt zur FA-Nummer wurde in der aktuellen Schicht schon erstellt und beinhaltet die Daten vom Test22
Abhängigkeiten	Test 22
Ablauf	- Retention Button drücken - Daten für Retention eingeben - Gewicht: 33 - Nass: 414 - Speichern Button drücken
Erwartete Resultate	Nach 1 sind die Felder Testperson mit ab, Produktions Code mit 158 und Zeit mit 12:00 ausgefüllt. Im Retention existiert nun 1 zusätzlicher Eintrag mit den Daten: - Prod Code: IT/11/16/158/12:00 - Gewicht trocken: 33 - Gewicht nass: 414 - Nach Zentrifuge (g): 381 - Nach Zentrifuge (%): 1154.55 - SAP Typ: EKX - SAP Nr. EN67 - g/Höwi 10.7 - Testperson: ab Die Retention Standardabweichung ist: - Gewicht nass: 14.85 - Nach Zentrifuge (g): 14.28 - Nach Zentrifuge (%): 24.08 - g/Höwi 0.44 Der Retention Durchschnitt ist: - Gewicht nass: 403.5 - Nach Zentrifuge (g): 370.9 - Nach Zentrifuge (%): 1137.52 - g/Höwi 10.39 Bei den Allgemeinen Werte sind nun die Werte verändert. Feld Retention für den neuen Eintrag, nach Zentrifuge (g) wurde grün markiert.

Test 24	
Funktionalität	Baby Windel Testblatt bearbeiten. Retention Test bearbeiten und Notizen hinzufügen und allgemeine Werte testen.
Vorbedingung	- Als LaborUser auf dem Testblatt mit der Fa-Nummer FA123456 - FA-Nummer FA123456 existiert und ist ein Baby Windel Artikel - Testblatt zur FA-Nummer wurde in der aktuellen Schicht schon erstellt und beinhaltet die Daten vom Test23
Abhängigkeiten	Test 23
Ablauf	- Retention Button unterster Edit Button drücken mit dem Testwert von Test 23. - Nass (g) anpassen mit Wert 300 - Notiz Button + drücken - Error Code 411 auswählen - Nachricht „2 cm Fehlen, habe die Maschine abgeschalten.“ hinzufügen. - Speichern Button drücken - Retention Button unterster Edit Button drücken mit dem Testwert von Test 23.
Erwartete Resultate	Nach 1 sind die Felder Testperson mit ab, Produktions Code mit 158, Zeit mit 12:00, Gewicht mit 33, Nass mit 414 ausgefüllt. Nach 6 wurde folgende Werte verändert bei der Datenzeile: - Gewicht nass: 300 - Nach Zentrifuge (g): 267 - Nach Zentrifuge (%): 809.09 - g/Höwi 7.21 Nach 6 haben sich alle Werte verändert bei der Retention Standardabweichung: - Gewicht nass: 65.76 - Nach Zentrifuge (g): 66.33 - Nach Zentrifuge (%):220.2 - g/Höwi 2.03 Nach 6 haben sich alle Werte verändert bei der Retention Durchschnitt: - Gewicht nass: 346.5 - Nach Zentrifuge (g): 313.9 - Nach Zentrifuge (%):964.79 - g/Höwi 8.65 Nach 6 wurde das Feld Retention für den bearbeiteten Eintrag, nach Zentrifuge (g) rot markiert. Nach 7 sind unter Notizen die Daten welche bei 4 und 5 eingeben wurden sichtbar.

Test 25	
Funktionalität	Baby Windel Testblatt bearbeiten. Retention Test löschen.
Vorbedingung	- Als LaborUser auf dem Testblatt mit der Fa-Nummer FA123456 - FA-Nummer FA123456 existiert und ist ein Baby Windel Artikel - Testblatt zur FA-Nummer wurde in der aktuellen Schicht schon erstellt und beinhaltet die Daten vom Test22 und Test24
Abhängigkeiten	Test 22, (Test 24)
Ablauf	1. Retention Button Löschen drücken mit dem Testwert von Test 22.
Erwartete Resultate	Testzeile wurde entfernt. Standardabweichung ist konstant 0. Durchschnittswerte sind gleich wie die einzelnen Testwerte des einzig existierenden Testwertes.

6.2.2.2 Inko Binden

Test 26	
Funktionalität	Inko Testblatt erstellen
Vorbedingung	- Als LaborUser auf der LaborCreator Startseite - FA-Nummer FA654321 existiert und ist ein Inko Artikel - Testblatt zur FA-Nummer wurde in der aktuellen Schicht noch nicht erstellt
Abhängigkeiten	Test 8
Ablauf	1. FA-Nummer FA654321 eingeben 2. Bearbeiten drücken
Erwartete Resultate	Weiterleitung auf Testblatt für Inko Binden mit FA-Nummer FA654321. Alle Werte für die Standardabweichung und den Durchschnitt sind 0. Es existieren keine Testdaten. Die Button Rewet, Aquisitionszeit und Retention existieren. Datum ist auf das aktuelle Datum gesetzt. Schicht ist auf die aktuelle Schicht gesetzt. Grösse und Produkt Text wurde von der Datenbank übernommen.

Test 27	
Funktionalität	Inko Testblatt bearbeiten. Rewet Test erstellen.
Vorbedingung	- Als LaborUser auf dem Testblatt mit der Fa-Nummer FA654321 - FA-Nummer FA654321 existiert und ist ein Inko Artikel - Testblatt zur FA-Nummer wurde in der aktuellen Schicht schon erstellt und ist leer
Abhängigkeiten	Test 26
Ablauf	- Rewet Button drücken - Daten für Rewet eingeben - Testperson: ab - Produktions Code: 158 - Zeit: 12:00 - EXP: 6/2019 - FP Trocken:12.9 - FP Feucht:12.91 - Speichern Button drücken
Erwartete Resultate	Im Rewet existiert nun 1 Eintrag mit den Daten: - Prod Code: IT/49/16/158/12:00 - FP trocken: 12.9 - FP Feucht:12.91 - Rewet: 0.01 - Testperson: ab Die Rewet Standardabweichung ist immer noch konstant 0. Der Rewet Durchschnitt ist: - FP trocken: 12.9 - FP Feucht:12.91 - Rewet: 0.01 Keine Felder wurden markiert.

Test 28	
Funktionalität	Inko Testblatt bearbeiten. Rewet Test erstellen. Standardabweichung testen.
Vorbedingung	- Als LaborUser auf dem Testblatt mit der Fa-Nummer FA654321 - FA-Nummer FA654321 existiert und ist ein Inko Artikel - Testblatt zur FA-Nummer wurde in der aktuellen Schicht schon erstellt und beinhaltet die Daten von Test 27
Abhängigkeiten	Test 27
Ablauf	- Rewet Button drücken - Daten für Rewet eingeben - FP Trocken:13.09 - FP Feucht:13.10 - Speichern Button drücken
Erwartete Resultate	Nach 1 sind die Daten ausgefüllt: - Testperson: ab - Produktions Code: 158 - Zeit: 12:00 - EXP: 6/2019 Im Rewet existiert nun 1 Eintrag mit den Daten: - Prod Code: IT/49/16/158/12:00 - FP trocken: 13.09 - FP Feucht: 13.10 - Rewet: 0.01 - Testperson: ab Die Rewet Standardabweichung ist: - FP trocken: 0.13 - FP Feucht:0.13 - Rewet: 0 Der Rewet Durchschnitt ist: - FP trocken: 13 - FP Feucht:13 - Rewet: 0.01 Keine Felder wurden markiert.

Test 29	
Funktionalität	Inko Testblatt bearbeiten. Aquisitionzeit Test erstellen.
Vorbedingung	- Als LaborUser auf dem Testblatt mit der Fa-Nummer FA654321 - FA-Nummer FA654321 existiert und ist ein Inko Artikel - Testblatt zur FA-Nummer wurde in der aktuellen Schicht schon erstellt und Aquisitionzeit ist leer.
Abhängigkeiten	Test 26
Ablauf	- Aquisitionzeit Button drücken - Daten für Aquisitionzeit eingeben - Testperson: ab - Produktions Code: 158 - Zeit: 12:00 - EXP: 6/2019 - Prüflingsgewicht: 21.15 1. Zugabe: 17.12 2. Zugabe: 54.06 3. Zugabe: 67.85 - FP Trocken: 21.73 - FP Feucht: 21.79 - Speichern Button drücken
Erwartete Resultate	Im Aquisitionszeit existiert nun 1 Eintrag mit den Daten: - Prod Code: IT/49/16/158/12:00 - Gewicht trocken: 21.15 1. Zugabe: 17.12 2. Zugabe: 54.06 3. Zugabe: 67.85 - FP Trocken: 21.73 - FP Feucht: 21.79 - Rewet: 0.06 - Testperson: ab Die Rewet Standardabweichung ist immer noch konstant 0. Der Rewet Durchschnitt ist: - Gewicht trocken: 21.15 1. Zugabe: 17.12 2. Zugabe: 54.06 3. Zugabe: 67.85 - FP Trocken: 21.73 - FP Feucht: 21.79 - Rewet: 0.06 Keine Felder wurden markiert.

Test 30	
Funktionalität	Inko Testblatt bearbeiten. Aquisitionzeit Test erstellen. Standardabweichung testen.
Vorbedingung	- Als LaborUser auf dem Testblatt mit der Fa-Nummer FA654321 - FA-Nummer FA654321 existiert und ist ein Inko Artikel - Testblatt zur FA-Nummer wurde in der aktuellen Schicht schon erstellt und Aquisitionzeit beinhaltet die Daten von Test 29.
Abhängigkeiten	Test 29 (Test 26)
Ablauf	- Aquisitionzeit Button drücken - Daten für Aquisitionzeit eingeben - Prüflingsgewicht: 29.78 1. Zugabe: 16.25 2. Zugabe: 53.64 3. Zugabe: 75.92 - FP Trocken: 21.57 - FP Feucht: 21.65 - Speichern Button drücken
Erwartete Resultate	Nach 1 sind die Daten ausgefüllt: - Testperson: ab - Produktions Code: 158 - Zeit: 12:00 - EXP: 6/2019 Im Aquisitionszeit existiert nun ein neuer Eintrag mit den Daten: - Prod Code: IT/49/16/158/12:00 - Gewicht trocken: 29.78 - 1. Zugabe: 16.25 - 2. Zugabe: 53.64 - 3. Zugabe: 75.92 - FP Trocken: 21.57 - FP Feucht: 21.65 - Rewet: 0.08 - Testperson: ab Die Rewet Standardabweichung ist: - Gewicht trocken: 61 - 1. Zugabe: 0.62 - 2. Zugabe: 0.3 - 3. Zugabe: 5.71 - FP Trocken: 0.11 - FP Feucht: 0.1 - Rewet: 0.01 Der Rewet Durchschnitt ist: - Gewicht trocken: 25.46 - 1. Zugabe: 16.69 - 2. Zugabe: 53.85 - 3. Zugabe: 71.88 - FP Trocken: 21.65 - FP Feucht: 21.72 - Rewet: 0.07 Keine Felder wurden markiert.

Test 31	
Funktionalität	Inko Testblatt bearbeiten. Retention Test erstellen.
Vorbedingung	- Als LaborUser auf dem Testblatt mit der Fa-Nummer FA654321 - FA-Nummer FA654321 existiert und ist ein Inko Artikel - Testblatt zur FA-Nummer wurde in der aktuellen Schicht schon erstellt und Retention ist leer.
Abhängigkeiten	Test 26
Ablauf	- Retention Button drücken - Daten für Retention eingeben - Testperson: ab - Produktions Code: 158 - Zeit: 12:00 - EXP: 6/2019 - Prüflingsgewicht: 30.21 - Prüfling nass: 430.15 - Prüfling nach Zentrifuge: 212.11 - Speichern Button drücken
Erwartete Resultate	Im Retention existiert nun 1 Eintrag mit den Daten: - Prod Code: IT/49/16/158/12:00 - Prüflingsgewicht: 30.21 - Prüfling nass: 430.15 - Prüfling nach Zentrifuge: 212.11 - Absorption 399.94 - Retention: 181.9 - Testperson: ab Die Retention Standardabweichung ist immer noch konstant 0. Der Retention Durchschnitt ist: - Prüflingsgewicht: 30.21 - Prüfling nass: 430.15 - Prüfling nach Zentrifuge: 212.11 - Absorption 399.94 - Retention: 181.9 Keine Felder wurden markiert.

Test 32	
Funktionalität	Inko Testblatt bearbeiten. Retention Test erstellen. Standardabweichung sowie Notiz hinzufügen und löschen testen.
Vorbedingung	- Als LaborUser auf dem Testblatt mit der Fa-Nummer FA654321 - FA-Nummer FA654321 existiert und ist ein Inko Artikel - Testblatt zur FA-Nummer wurde in der aktuellen Schicht schon erstellt und Retention beinhaltet die Daten von Test 31.
Abhängigkeiten	Test 31 (Test 26)
Ablauf	- Retention Button drücken - Daten für Retention eingeben - Prüflingsgewicht: 20.47 - Prüfling nass: 437.64 - Prüfling nach Zentrifuge: 216.43 - Notizen Button drücken - Error Code 080 auswählen - Nachricht: „eifach weg isches“ erfassen. - Speichern Button drücken - Auf den Editieren Button für den vorhin erstellten Retention Test drücken - Löschen Button bei Notiz drücken - Speichern Button drücken - Auf den Editieren Button für den vorhin erstellten Retention Test drücken
Erwartete Resultate	Nach 1 sind die Daten ausgefüllt: - Testperson: ab - Produktions Code: 158 - Zeit: 12:00 - EXP: 6/2019 Nach 6 sind folgende Daten sichtbar: Im Retention existiert nun ein neuer Eintrag mit den Daten: - Prod Code: IT/49/16/158/12:00 - Prüflingsgewicht: 20.47 - Prüfling nass: 437.64 - Prüfling nach Zentrifuge: 216.43 - Absorption 417.17 - Retention: 195.96 - Testperson: ab Die Retention Standardabweichung ist: - Prüflingsgewicht: 6.89 - Prüfling nass: 5.3 - Prüfling nach Zentrifuge: 3.05 - Absorption 12.18 - Retention: 9.94 - Testperson: ab Der Retention Durchschnitt ist: - Prüflingsgewicht: 25.34 - Prüfling nass: 433.9 - Prüfling nach Zentrifuge: 214.27 - Absorption 408.55 - Retention: 188.93 Keine Felder wurden markiert.

Nach 7 sind unter Notizen die eingegebenen Notizen sichtbar
 Nach 10 sind unter Notizen die Notizen nicht mehr sichtbar, da sie gelöscht sind.

Test 33	
Funktionalität	Inko Testblatt bearbeiten. Retention Test löschen.
Vorbedingung	- Als LaborUser auf dem Testblatt mit der Fa-Nummer FA654321 - FA-Nummer FA654321 existiert und ist ein Inko Artikel - Testblatt zur FA-Nummer wurde in der aktuellen Schicht schon erstellt und Retention beinhaltet die Daten von Test 32.
Abhängigkeiten	Test 32
Ablauf	1. Lösch Button bei Retention für die zweite Testzeile mit den Testdaten von Test 32 drücken.
Erwartete Resultate	Testzeile wurde entfernt. Standardabweichung ist konstant 0. Durchschnittswerte sind gleich wie die einzelnen Testwerte des einzig existierenden Testwertes.

6.2.3 Dashboard

Test 34	
Funktionalität	Dashboard Default Ansicht.
Vorbedingung	- Als LaborUser auf dem Dashboard - Tests 17-33 wurden am aktuellen Tag durchgeführt.
Abhängigkeiten	Test 17-33 (Daten), Test 11 (Berechtigung)
Ablauf	-
Erwartete Resultate	In Spalte der Maschine M11 ist das Testblatt FA123456 sichtbar, welches einen roten Rand hat. Sowie ein Icon mit der Zahl 1 für die Anzahl der Notizen und ein Link mit dem ø-Zeichen. In Spalte der Maschine M49 ist das Testblatt FA654321 sichtbar, welches einen grünen Rand hat. Sowie ein Link mit dem ø-Zeichen.

Test 35	
Funktionalität	Dashboard Durchschnittswerte M11.
Vorbedingung	- Als LaborUser auf dem Dashboard - Tests 17-33 wurden am aktuellen Tag durchgeführt.
Abhängigkeiten	Test 17-33 (Daten), Test 11 (Berechtigung)
Ablauf	1. Auf ø-Zeichen von FA123456 in Zeile M11 klicken
Erwartete Resultate	Folgende Werte sind sichtbar: - Retention - Nach Zentrifuge (g): 267 - Rewet - 140 ml: 0.15 - Rewet - 210 ml: 0.6 - Rewet - 140 ml: 0 - Rewet - 210 ml: 0 - Penetrationszeit - Zugabe 4: 170 Zeile „Retention - Nach Zentrifuge (g)“ und „Rewet - 210 ml“ sind rot markiert. Zeile „Penetrationszeit - Zugabe 4“ ist orange markiert.

Test 36	
Funktionalität	Dashboard Durchschnittswerte M49.
Vorbedingung	- Als LaborUser auf dem Dashboard - Tests 17-33 wurden am aktuellen Tag durchgeführt.
Abhängigkeiten	Test 17-33 (Daten), Test 11 (Berechtigung)
Ablauf	1. Auf ø-Zeichen von FA654321 in Zeile M49 klicken
Erwartete Resultate	Folgende Werte sind sichtbar: - Aquisitionszeit - Zugabe 1: 16.69 - Aquisitionszeit - Zugabe 2: 53.85 - Aquisitionszeit - Zugabe 3: 71.88 - Rewet nach Aquisition: 0.07 - Retention: 181.9 - Rewet: 0.01 Keine Zeile ist markiert.

Test 37	
Funktionalität	Dashboard Notizwerte.
Vorbedingung	- Als LaborUser auf dem Dashboard - Tests 17-33 wurden am aktuellen Tag durchgeführt.
Abhängigkeiten	Test 17-33 (Daten), Test 11 (Berechtigung)
Ablauf	1. Auf Notiz-Zeichen von FA123456 in Zeile M11 klicken
Erwartete Resultate	Folgende Werte sind sichtbar: - Code: 411 - Meldung: Saugkissen hinten zu kurz - Nachricht: 2 cm Fehlen, habe Maschine abgeschalten.

Test 38	
Funktionalität	Dashboard Verlinkung auf Baby Windel Testblatt mit LaborUser
Vorbedingung	- Als LaborUser auf dem Dashboard - Tests 17-33 wurden am aktuellen Tag durchgeführt.
Abhängigkeiten	Test 17-33 (Daten), Test 11 (Berechtigung)
Ablauf	1. Box FA123456 in Zeile M11 klicken
Erwartete Resultate	Weiterleitung auf Testblatt von FA123456 des Datums und Schicht zu welcher es gehört. Gleiche Ansicht wie bei LaborCreator.

Test 39	
Funktionalität	Dashboard Verlinkung auf Inko Binden Testblatt mit LaborUser
Vorbedingung	- Als LaborUser auf dem Dashboard - Tests 17-33 wurden am aktuellen Tag durchgeführt.
Abhängigkeiten	Test 17-33 (Daten), Test 11 (Berechtigung)
Ablauf	1. Box FA654321 in Zeile M49 klicken
Erwartete Resultate	Weiterleitung auf Testblatt von FA654321 des Datums und Schicht zu welcher es gehört. Gleiche Ansicht wie bei LaborCreator.

Test 40	
Funktionalität	Dashboard Verlinkung auf Baby Windel Testblatt mit LaborViewer
Vorbedingung	- Als LaborViewer auf dem Dashboard - Tests 17-33 wurden am aktuellen Tag durchgeführt.
Abhängigkeiten	Test 17-33 (Daten), Test 12 (Berechtigung)
Ablauf	1. Box FA123456 in Zeile M11 klicken
Erwartete Resultate	Weiterleitung auf Testblatt von FA123456 des Datums und Schicht zu welcher es gehört. Gleiche Ansicht wie bei LaborCreator. Aber ohne die Buttons für das Erstellen von Rewet und Retention Test und ohne die Zeilen die das Editieren und Löschen der einzelnen Testzeilen ermöglicht.

Test 41	
Funktionalität	Dashboard Verlinkung auf Inko Binden Testblatt mit LaborViewer
Vorbedingung	• Als LaborViewer auf dem Dashboard • Tests 17-33 wurden am aktuellen Tag durchgeführt.
Abhängigkeiten	Test 17-33 (Daten), Test 12 (Berechtigung)
Ablauf	1. Box FA654321 in Zeile M49 klicken
Erwartete Resultate	Weiterleitung auf Testblatt von FA654321 des Datums und Schicht zu welcher es gehört. Gleiche Ansicht wie bei LaborCreator. Aber ohne die Buttons für das Erstellen von Rewet. Aquisitionzeit und Retention Test und ohne die Zeilen die das Editieren und Löschen der einzelnen Testzeilen ermöglicht.

6.2.4 History

Test 42	
Funktionalität	History Suchfunktion Fehlerhafte Eingaben
Vorbedingung	- Als User auf der History - Test 17-33 (Daten)
Abhängigkeiten	Test 17-33 (Daten)
Ablauf	- Falsche FaNummer eingeben „fa“ eingeben „fa23“ eingeben „13357“ eingeben „FA“ eingeben und Bearbeiten drücken „FA5“ eingeben und Bearbeiten drücken - Feld leer lassen und Bearbeiten drücken „asdf“ eingeben und Bearbeiten drücken
Erwartete Resultate	1a, 1b, 1c, Fehlermeldung wird angezeigt: „Fertigungsnummer muss mit FA beginnen und mit Nummern Enden.“ 1d, Fehlermeldung wird angezeigt: „Fertigungsnummer muss mindestens 3 Zeichen lang sein.“ 1e, Fehlermeldung wird angezeigt: „Kein Eintrag mit Fertigungsnummer FA5 gefunden.“ 1f, Fehlermeldung wird angezeigt: „Kein Eintrag mit Fertigungsnummer gefunden.“ 1g, Fehlermeldung wird angezeigt: „Fertigungsnummer muss mit FA beginnen und mit Nummern Enden.“ und Bearbeiten Button löst nichts aus.

Test 43	
Funktionalität	History Suchfunktion History nach FA-Nummer durchsuchen
Vorbedingung	- Als User auf der History - Test 17-26 (Daten)
Abhängigkeiten	Test 17-33 (Daten)
Ablauf	FA-Nummer „FA123456“ eingeben
Erwartete Resultate	Eintrag für den aktuellen Tag und der Schicht bei welcher die FA123456 beim Test 18 hatte. Ansicht hat einen roten Rand und beinhaltet die Daten: FA123456 - M11 – {Datum} – {Schicht}

Test 44	
Funktionalität	Von der History zum Testblatt
Vorbedingung	- Als User auf der History - Test 43 wurde bis zum Schluss durchgeführt
Abhängigkeiten	Test 43
Ablauf	Auf bei Test43 beschriebene Zeile klicken
Erwartete Resultate	Weiterleitung zum Testblatt

7 Zusammenfassung

7.1 Ergebnisse

Das Ergebnis dieser Studienarbeit ist eine Intranet-Applikation für die Firma Intigena, welches durch Module erweiterbar ist. Das Intranet beinhaltet neben der Shell ein Labor Modul.

Die Shell, welches die Rahmenapplikation für das Intranet ist, wurde als ASP.NET MVC 5 Applikation entwickelt nach dem MVVM-Pattern. Als Datenbank kommt ein SQL Server 2014 zum Einsatz.

Das Labor Modul wurde ebenfalls als ASP.Net MVC 5 Applikation nach dem MVVM-Pattern entwickelt. Das Backend des Labors befindet sich in einem eigenständigen Projekt. Das Frontend des Labors, die Controller und die Views, befindet sich jedoch im Webprojekt der Shell.

Das ganze Projekt wurde nach dem Inversion of Control Prinzip entwickelt.

Damit das Intranet einfach zu benutzen ist, wurden alle Seiten in einem schlichten, einheitlichen Design erstellt, welche keine überflüssigen Informationen enthalten.

Um das Intranet sowie das Labor-Modul zu testen wurde ein Testplan erstellt. Für den Business Layer wurden Unit-Tests geschrieben, welche die Funktionalität testen.

7.2 Ausblick

7.2.1 Erweiterungsmöglichkeiten Shell

Dieses Kapitel beschreibt einige Erweiterungsmöglichkeiten für die Shell, welche zu einem späteren Zeitpunkt hinzugefügt werden können.

7.2.1.1 Weitere Module

Das Intigena Intranet könnte in Zukunft noch mit weiteren Modulen erweitert werden. Bei der Einführungsbesprechung für diese Studienarbeit waren mögliche Modul-Erweiterungen die folgenden:

- Schichtplan
- Ersatzteilkverwaltung (Lagerverwaltung / Bestellungen)
- Maschinen Info (Effizienz etc.)

7.2.2 Erweiterungsmöglichkeiten Labor-Modul

Dieses Kapitel beschreibt einige Erweiterungsmöglichkeiten für das Labor Modul, welche zu einem späteren Zeitpunkt hinzugefügt werden können.

7.2.2.1 Tampon

Bis jetzt wurden nur Babywindel und Inkonsistenz-Windel Tests im Modul beachtet. Da aber im Labor auch noch Tampons getestet werden, könnte das Modul noch mit den Tampon-Testdurchführungen erweitert werden.

7.2.2.2 XML-Parser

Im Labor der Intigena AG gibt es zu den menschlichen Testdurchführungen auch noch maschinelle Tests durch eine Instron-Maschine. Diese Maschine generiert für jeden Test ein PDF und ein XML-File mit Testresultaten. Das XML-File müsste noch geparkt werden um die einzelnen Testwerte in die Datenbank zu laden. Zusätzlich sollten vor oder nach dem Parsen und in die Datenbank schreiben die von der Maschine generierten Files an einen anderen Speicherort verschoben werden. Im Verlaufe dieser Studienarbeit wurde bereits eine Anforderungsanalyse für den XML-Parser durchgeführt.

7.3 Verbesserungsvorschläge

7.3.1 Server Side Caching

Momentan beinhalten alle Requests mindestens einen Datenbankzugriff für die Navigation. Beim Abfüllen der Dashboard-View beispielsweise gibt es sogar mehrere Datenbankzugriffe um alle Informationen zu holen, was zu geringen Performanz Einbussen führt. Dies könnte durch Caching von Daten auf dem Server optimiert werden. Die Datenbankzugriffe für die Schichtplan Informationen würden sich dafür zum Beispiel gut eignen, da sich diese Informationen sehr selten Ändern.

7.3.2 Navigation

In der Navigation gibt es auch noch Verbesserungsvorschläge. Bis jetzt werden in der Navigation nur alle Module aufgelistet. Module könnten aber unter Umständen selber kleine Module beinhalten, wie das Labor Modul, welches aus einem Dashboard, einer Testerfassung und einer History besteht. Diese Untermodule könnten in die Navigation eingebaut werden. Dadurch würden für Benutzer unnötige Userinteraktionen eingespart werden, da sie direkt auf ihren Zielbereich gelangen könnten, anstatt erst über das Modul zu navigieren. Diese Erweiterungsmöglichkeiten gestalten sich durch unser sauberes Navigationsdesign sehr einfach.

8 Verzeichnisse

8.1 Glossar und Abkürzungsverzeichnis

Inko	Bezeichnung von Produkten für die Inkontinenzhygiene
Rewet	Name für eine Testart, welche für Inko- und Babywindeln durchgeführt wird. Kommt aus dem Englischen (to wet again)
Retention	Name für eine Testart, welche für Inko- und Babywindeln durchgeführt wird. Stammt aus dem Lateinischen (lat.: retinere = zurückhalten)

8.2 Abbildungen

Abbildung 1: MVC simple.....	8
Abbildung 2: MVVM	8
Abbildung 3: MVC + MVVM	8
Abbildung 4: Schichtenarchitektur.....	9
Abbildung 5: Final MVC Architektur	9
Abbildung 6: DIP, IoC und DI	10
Abbildung 7: Code Map von Home Service via Visual Studio	11
Abbildung 8: Abhängigkeiten	14
Abbildung 9: Schichtenmodell Version 1.....	16
Abbildung 10: Schichtenmodell Version 2.....	16
Abbildung 11: Schichtenmodell Version 3.....	17
Abbildung 12: Rollenbeispiel für die Autorisation	20
Abbildung 13: Labor Controller mit Attribut.....	25
Abbildung 14: Klassendiagramm Shell Version 1.....	26
Abbildung 15: TBH Skizze	27
Abbildung 16: TBT Skizze.....	27
Abbildung 17: TPC Skizze	28
Abbildung 18: DB Diagramm Shell Version 1.....	28
Abbildung 19: DB Diagramm Shell Version 2.....	29
Abbildung 20: DB Daten Shell Modules	30
Abbildung 21: DB Daten Shell Roles.....	30
Abbildung 22: Klassendiagramm GenericRepository	31
Abbildung 23: Klassendiagramm GenericRepository 2	31
Abbildung 24: Klassendiagramm Navigation.....	34
Abbildung 25: Klassendiagramm LoggingBase.....	34
Abbildung 26: Klassendiagramm Logging.....	36
Abbildung 27: Dependencydiagram Shell	37
Abbildung 28: Intigena Webseite www.intigena.ch	38

Abbildung 29: Mockup der Navigationsleiste.....	38
Abbildung 30: Endergebnis der Navigationsleiste	38
Abbildung 31: Mockup der Intranet Startseite	39
Abbildung 32: Endergebnis der Intranet Startseite	39
Abbildung 33: Mockup der Einstellungsseite.....	39
Abbildung 34: Endergebnis der Einstellungsseite	39
Abbildung 35: Endergebnis der Startseite angezeigt auf einem Samsung S5	39
Abbildung 36: Skizze Labor Datenbank	41
Abbildung 37: Datenbankdiagramm Labor	42
Abbildung 38: Skizze Intigena Stammdaten Datenbank.....	43
Abbildung 39: Datenbankdiagramm Intigena Schicht.....	44
Abbildung 40: Stammdaten Schicht	44
Abbildung 41: Datenbankdiagramm Intigena Stammdaten	45
Abbildung 42: Maschine Datenbeispiel	45
Abbildung 43: Artikel Datenbeispiel	46
Abbildung 44: Fertigungsartikelkomponente Datenbeispiel.....	46
Abbildung 45: Fertigungsartikel Datenbeispiel	47
Abbildung 46: Skizze Datenbank Instron	47
Abbildung 47: Datenbankdiagramm Instron	48
Abbildung 48: Dependency Diagramm Labor.....	49
Abbildung 49: Sequenzdiagramm für den Aufruf eines Inko-Rewet Testformulars.....	50
Abbildung 50: Sequenzdiagramm für das Speichern eines Inko-Rewet Testformulars.....	51
Abbildung 51: Startseite des Labor-Modul	52
Abbildung 52: Dashboard Wireframe	52
Abbildung 53: Detailansicht im Dashboard.....	52
Abbildung 54: Endergebnis des Dashboards	53
Abbildung 55: Endergebnis der Detailansicht im Dashboard.....	53
Abbildung 56: Wireframe zur Startseite für die Testerfassung.....	53
Abbildung 57: Endergebnis der Startseite für die Testerfassung.....	53
Abbildung 58: Wireframe der Detailansicht für Babywindeln	54
Abbildung 59: Endergebnis der Babywindeln.....	55
Abbildung 60: Wireframe für die Rewet-Testeingabe	55
Abbildung 61: Endergebnis des Rewet-Testeingabe-Formulars.....	55
Abbildung 62: Ablaufdiagramm aus der Sicht eines Benutzers	56
Abbildung 67: Test-Coverage aus dem Visual Studio	89
Abbildung 68: Vom Visual Studio generierte Code Metrics	89
Abbildung 69: Testzusammenfassung vom Build.....	90
Abbildung 70: Dependency Diagramm Applikation	94
Abbildung 71: Package Diagramm Shell.....	95
Abbildung 72: Package Diagramm Labor.....	96
Abbildung 73: Klassendiagramm ganze Shell.....	97
Abbildung 74: Klassendiagramm Common Übersicht	98
Abbildung 75: Klassendiagramm Common Detail	99
Abbildung 76: Klassendiagramm Intranet.Common Bll	100
Abbildung 77: Klassendiagramm Intranet.Common Db.....	101
Abbildung 78: Klassendiagramm Intranet.Common Disposable.....	102
Abbildung 79: Klassendiagramm Intranet.Common Logging.....	102

Abbildung 80: Klassendiagramm Intranet.Common Role	103
Abbildung 81: Klassendiagramm Intranet.Common Web	103
Abbildung 82: Klassendiagramm Intranet.Definition	104
Abbildung 83: Klassendiagramm Intranet.Bll Übersicht	104
Abbildung 84: Klassendiagramm Intranet.Bll Detailliert	105
Abbildung 85: Klassendiagramm Intranet.Dal	106
Abbildung 86: Klassendiagramm Intranet.Dal detailliert	106
Abbildung 87: Klassendiagramm Model Shell	107
Abbildung 88: Klassendiagramm ViewModel Shell	107
Abbildung 89: Klassendiagramm Intranet.Web Übersicht	109
Abbildung 90: Klassendiagramm Intranet.Web Detail	110
Abbildung 91: Model Fertigungsartikel	111
Abbildung 92 Model Labordaten	112
Abbildung 93: Model Fertigungsartikel	113
Abbildung 94: Viewmodel Instron	114
Abbildung 95: Model Schichtplan	114
Abbildung 96: Klassendiagramm Labor.Bll Übersicht	115
Abbildung 97: Klassendiagramm Labor.Bll Detail Teil1	116
Abbildung 98: Klassendiagramm Labor.Bll Detail Teil 2	117
Abbildung 99: Klassendiagramm Labor.Bll Detail Teil 3	118
Abbildung 100: Klassendiagramm Labor.Bll Detail Teil 4	119
Abbildung 101: Klassendiagramm Labor.Bll Detail Teil 5	120
Abbildung 102: Klassendiagramm Labor.Bll Detail Teil 6	121
Abbildung 103: Klassendiagramm Labor.Bll Detail gesamt	121
Abbildung 104: Klassendiagramm Labor.Dal	122
Abbildung 105: Klassendiagramm Labor.Definition Übersicht	123
Abbildung 106: Klassendiagramm Labor.Definition Detail Teil 1	124
Abbildung 107: Klassendiagramm Labor.Definition Detail Teil 2	125
Abbildung 108: Klassendiagramm Labor.ViewModel Übersicht	126
Abbildung 109: Klassendiagramm Labor.ViewModel Detail	127

8.3 Tabellen

Tabelle 1: Routing Übersicht.....	20
Tabelle 2: DB Shell Werte Beschreibung	30
Tabelle 3: DB Daten Shell Roles Werte Beschreibung.....	30
Tabelle 4: Schicht DB Werte Beschreibung	44
Tabelle 5: Maschine DB Werte Beschreibung.....	45
Tabelle 6: InstronValues Werte Beschreibung	48
Tabelle 7: Instrons Werte Beschreibung	48
Tabelle 8: Ergebnisse vom Systemtest 1 - 12.12.16 – Firefox.....	91
Tabelle 9: Ergebnisse vom Systemtest 2 - 12.12.16 – Firefox.....	92
Tabelle 10: Ergebnisse vom Systemtest 3 - 12.12.16 - IE.....	93
Tabelle 11: ViewModel Shell Werte Beschreibung.....	108

8.4 Literatur und Quellenverzeichnis

- [1] D. Team, «Devart Blog,» [Online]. Available: <http://blog.devart.com/table-per-type-vs-table-per-hierarchy-inheritance.html>. [Zugriff am 10 13 2016].
- [2] T. Dykstra, «Implementing Inheritance with the Entity Framework 6 in an ASP.NET MVC 5 Application,» Microsoft, 7 11 2014. [Online]. Available: <https://www.asp.net/mvc/overview/getting-started/getting-started-with-ef-using-mvc/implementing-inheritance-with-the-entity-framework-in-an-asp-net-mvc-application>. [Zugriff am 10 13 2016].
- [3] Microsoft, «EF Designer TPH Inheritance,» Microsoft, [Online]. Available: <https://msdn.microsoft.com/en-us/data/jj618292.aspx>. [Zugriff am 13 10 2016].
- [4] Microsoft, «EF Designer TPT Inheritance,» Microsoft, [Online]. Available: <https://msdn.microsoft.com/en-us/data/jj618293.aspx>. [Zugriff am 13 10 2016].
- [5] Microsoft, «Managed Extensibility Framework (MEF),» Microsoft, [Online]. Available: [https://msdn.microsoft.com/de-ch/library/dd460648\(v=vs.110\).aspx](https://msdn.microsoft.com/de-ch/library/dd460648(v=vs.110).aspx). [Zugriff am 25 11 2016].
- [6] H. Haque, «Fitting altogether (DIP, IoC and DI),» code project, [Online]. Available: https://www.codeproject.com/KB/architecture/538536/IOC2_v1.jpg,. [Zugriff am 05 12 2016].
- [7] «xunit,» xunit, [Online]. Available: <https://xunit.github.io/>. [Zugriff am 05 12 2016].
- [8] «Moq,» [Online]. Available: <https://github.com/Moq/moq4/wiki/Quickstart>. [Zugriff am 05 12 2016].
- [9] H. Haque, "A curry of Dependency Inversion Principle (DIP), Inversion of Control (IoC), Dependency Injection (DI) and IoC Container," code project, [Online]. Available: <http://www.codeproject.com/Articles/538536/A-curry-of-Dependency-Inversion-Principle-DIP-Inve>. [Accessed 24 11 2016].

9 Danksagung

An dieser Stelle möchten wir uns bei all denjenigen bedanken, die uns während dieser Studienarbeit unterstützt und motiviert haben.

Wir möchten unseren Dank an folgenden Personen aussprechen:

- Herrn Jürg Jucker von S3CC GmbH, Pfäffikon, für die Betreuung der Studienarbeit sowie für die hilfreichen Anregungen und konstruktiven Kritiken
- Prof. Hansjörg Huser vom Institute for Networked Solutions (INS), HSR Rapperswil, für die Anfängliche Betreuung der Studienarbeit
- Unseren Familien und Freunden für die tatkräftige Unterstützung während unserer Semesterarbeit

10 Anhang

10.1 Projekt Management

10.1.1 Code Auswertungen

10.1.1.1 Test-Coverage

Die Test-Coverage in dieser Semesterarbeit durch Unit Tests beträgt ~87%. Alle Unit Tests laufen erfolgreich ab wie im Testbericht ersichtlich ist.

Hierarchy	Not Covered (Blocks)	Not Covered (% Blocks)	Covered (Blocks)	Covered (% Blocks) ▼
▲ f Jordi_PIN1262012 2016-12-07 11_34_5...	2481	13.08%	16491	86.92%
▶ intranet.bll.test.dll	0	0.00%	488	100.00%
▶ intranet.model.dll	0	0.00%	26	100.00%
▶ intranet.model.test.dll	0	0.00%	91	100.00%
▶ intranet.viewmodel.dll	0	0.00%	14	100.00%
▶ intranet.viewmodel.test.dll	0	0.00%	225	100.00%
▶ intranet.labor.testenvironment.dll	6	0.26%	2268	99.74%
▶ intranet.labor.bll.test.dll	71	0.93%	7550	99.07%
▶ intranet.web.test.dll	18	4.20%	411	95.80%
▶ intranet.testenvironment.dll	42	10.58%	355	89.42%
▶ intranet.labor.viewmodel.dll	45	12.64%	311	87.36%
▶ intranet.labor.bll.dll	948	18.21%	4258	81.79%
▶ intranet.labor.model.dll	63	22.50%	217	77.50%
▶ intranet.bll.dll	179	63.93%	101	36.07%
▶ intranet.web.dll	749	83.13%	152	16.87%
▶ intranet.common.dll	293	92.43%	24	7.57%

Abbildung 63: Test-Coverage aus dem Visual Studio

10.1.1.2 Code Metrics

Hierarchy ▼	Maintainability Index	Cyclomatic Complexity	Depth of Inheritance	Class Coupling	Lines of Code
▶ Labor\Intranet.Labor.ViewModel (Debug)	94	390	2	38	390
▶ Labor\Intranet.Labor.Model (Debug)	95	294	2	29	294
▶ Labor\Intranet.Labor.Dal (Debug)	88	38	4	34	184
▶ Labor\Intranet.Labor.Bll (Debug)	70	794	3	171	1'762
▶ Labor\Test\Intranet.Labor.TestEnvironment (Debug)	64	86	1	82	352
▶ Labor\Test\Intranet.Labor.Bll.Test (Debug)	64	488	1	211	2'145
▶ Intranet.Web (Debug)	82	169	4	175	344
▶ Intranet.ViewModel (Debug)	94	18	1	3	18
▶ Intranet.Model (Debug)	97	29	2	11	29
▶ Intranet.Dal (Debug)	84	14	4	29	29
▶ Intranet.Common (Debug)	91	148	3	58	155
▶ Intranet.Bll (Debug)	77	41	3	42	145
▶ _Test\Intranet.Web.Test (Debug)	71	51	1	37	163
▶ _Test\Intranet.ViewModel.Test (Debug)	74	27	1	24	54
▶ _Test\Intranet.TestEnvironment (Debug)	66	27	1	46	126
▶ _Test\Intranet.Model.Test (Debug)	77	8	1	16	22
▶ _Test\Intranet.Integrationtest (Debug)	78	10	2	15	26
▶ _Test\Intranet.Bll.Test (Debug)	67	24	1	36	158

Abbildung 64: Vom Visual Studio generierte Code Metrics

10.1.1.2.1 Lines of Code

Das Projekt besteht aus 6'398 Zeilen C# (.cs) Code ohne Kommentare und Leerzeilen von denen 3'260 Zeilen produktiver Code und 3138 Zeilen Testcode sind. Zusätzlich besteht das Projekt noch aus 1'652

Zeilen View Code (.cshtml), wovon 1526 Zeilen von den Labor Views sind und 126 Zeilen von der Shell, sowie 260 Zeilen selbstgeschriebenen SASS Code und 75 Zeilen selbstgeschriebenen TypeScript Code.

10.1.1.2.2 Cyclomatic Complexity

In Abbildung 64 werden alle Werte für die einzelnen Projekte summiert angezeigt. Im Durchschnitt liegt die Cyclomatic Complexity für die einzelnen Methoden zwischen 1 und 5 was sehr gut ist. Es gibt aber wenige Methoden, bei denen der Wert ~30 beträgt. Der Grund dafür ist, dass diese Methoden für die Entscheidung ob die Testwerte gut oder schlecht sind, was einige if else abfragen beinhaltet.

10.2 Testbericht

10.2.1 Unit Test

Es gibt insgesamt 343 automatisierte Unit Tests. 28 davon sind Controllertests und 285 testen die Funktionalität des Labormodules. Die Durchführung der Unit Tests dauert bisschen weniger als eine Sekunde. Alle Tests laufen auf dem Entwicklersystem sowie auf dem Build Server erfolgreich ab.

```

=== TEST EXECUTION SUMMARY ===
Intranet.Bll.Test                Total: 19, Errors: 0, Failed: 0, Skipped: 0, Time: 0.110s
Intranet.Common.Test.dll         Total: 0
Intranet.Dal.Test.dll            Total: 0
Intranet.Labor.Bll.Test          Total: 285, Errors: 0, Failed: 0, Skipped: 0, Time: 0.371s
Intranet.Labor.Dal.Test.dll       Total: 0
Intranet.Labor.Model.Test.dll     Total: 0
Intranet.Labor.ViewModel.Test.dll Total: 0
Intranet.ViewModel.Test          Total: 11, Errors: 0, Failed: 0, Skipped: 0, Time: 0.032s
Intranet.Web.Test                Total: 28, Errors: 0, Failed: 0, Skipped: 0, Time: 0.387s
                                ---
                                GRAND TOTAL: 343          0          0          0          0.900s
  
```

Abbildung 65: Testzusammenfassung vom Build

10.2.2 Usability Test

Der Usability Test wurde aus zeitlichen Gründen abgesagt. Nach hinten verschieben machte für die Arbeit keinen Sinn mehr, da die Erkenntnisse aus dem Test nicht mehr in die Arbeit hätte hineinfließen können.

10.2.3 System Test

10.2.3.1 Ergebnisse vom Systemtest 1 – 12.12.16 – Firefox

	Passed	Failed	Bemerkung
Test 1	X		
Test 2	X		
Test 3	X		
Test 4	X		
Test 5	X		
Test 6	X		
Test 7	X		
Test 8	X		
Test 9	X		
Test 10	X		
Test 11	X		
Test 12	X		
Test 13	X		
Test 14	X		
Test 15	X		

Test 16	X		
Test 17	X		
Test 18	X		
Test 19	X		
Test 20		X	Zahlen mit Kommastellen werden als 0 abgespeichert
Test 21		X	Zahlen mit Kommastellen werden als 0 abgespeichert
Test 22		X	Zahlen mit Kommastellen werden als 0 abgespeichert
Test 23		X	Zahlen mit Kommastellen werden als 0 abgespeichert
Test 24		X	Zahlen mit Kommastellen werden als 0 abgespeichert
Test 25		X	Zahlen mit Kommastellen werden als 0 abgespeichert
Test 26		X	Zahlen mit Kommastellen werden als 0 abgespeichert
Test 27		X	Zahlen mit Kommastellen werden als 0 abgespeichert
Test 28		X	Zahlen mit Kommastellen werden als 0 abgespeichert
Test 29		X	Zahlen mit Kommastellen werden als 0 abgespeichert
Test 30		X	Zahlen mit Kommastellen werden als 0 abgespeichert
Test 31		X	Zahlen mit Kommastellen werden als 0 abgespeichert
Test 32		X	Zahlen mit Kommastellen werden als 0 abgespeichert
Test 33		X	Zahlen mit Kommastellen werden als 0 abgespeichert
Test 34		X	Zahlen mit Kommastellen werden als 0 abgespeichert
Test 35		X	Zahlen mit Kommastellen werden als 0 abgespeichert
Test 36		X	Zahlen mit Kommastellen werden als 0 abgespeichert
Test 37		X	Zahlen mit Kommastellen werden als 0 abgespeichert
Test 38		X	Zahlen mit Kommastellen werden als 0 abgespeichert
Test 39		X	Zahlen mit Kommastellen werden als 0 abgespeichert
Test 40		X	Zahlen mit Kommastellen werden als 0 abgespeichert
Test 41		X	Zahlen mit Kommastellen werden als 0 abgespeichert
Test 42		X	Zahlen mit Kommastellen werden als 0 abgespeichert
Test 43		X	Zahlen mit Kommastellen werden als 0 abgespeichert
Test 44		X	Zahlen mit Kommastellen werden als 0 abgespeichert

Tabelle 8: Ergebnisse vom Systemtest 1 - 12.12.16 – Firefox

Bei diesem Systemtest sind beim Test 20 Fehler aufgetreten. Es konnten keine Kommazahlen abgespeichert werden. Das Resultat einer abgespeicherten Kommazahl war immer Null. Alle darauffolgenden Tests sind ebenfalls fehlgeschlagen, da diese entweder auch Kommazahlen speichern mussten oder eine Abhängigkeit von einem bereits fehlgeschlagenen Test gehabt haben.

Der Grund für diesen Fehler war, dass keine CultureInfo auf dem Server definiert wurden und dieser nur Kommazahlen mit einem Komma und nicht mit einem Punkt akzeptierte.

Dieser Fehler wurde behoben und es wurde daraufhin ein weiterer Systemtest mit Firefox durchgeführt.

10.2.3.2 Ergebnisse vom Systemtest 2 – 12.12.16 – Firefox

	Passed	Failed	Bemerkung
Test 1	X		
Test 2	X		
Test 3	X		
Test 4	X		
Test 5	X		
Test 6	X		
Test 7	X		

Test 8	X		
Test 9	X		
Test 10	X		
Test 11	X		
Test 12	X		
Test 13	X		
Test 14	X		
Test 15	X		
Test 16	X		
Test 17	X		
Test 18	X		
Test 19	X		
Test 20	X		Standardabweichung NaN anstatt n. def
Test 21	X		
Test 22	X		
Test 23	X		
Test 24	X		
Test 25	X		
Test 26	X		
Test 27	X		
Test 28	X		
Test 29	X		StDev + ø werden nicht fett geschrieben
Test 30	X		
Test 31	X		StDev + ø werden für Retention und Absorption nicht fett geschrieben
Test 32	X		
Test 33	X		
Test 34	X		
Test 35	X		
Test 36	X		
Test 37	X		
Test 38	X		
Test 39	X		
Test 40	X		
Test 41	X		
Test 42	X		
Test 43	X		
Test 44	X		

Tabelle 9: Ergebnisse vom Systemtest 2 - 12.12.16 – Firefox

Beim zweiten Systemtest mit Mozilla Firefox konnte jeder Test erfolgreich durchgeführt werden.

Beim Test 20 gab es einen Unterschied bei der Anzeige einer nicht definierten Zahl, welches auf eine andere Version im Server zurückzuführen ist.

Beim Test 29 sowie 31 fiel auf, dass gewisse Textabschnitte, die fett geschrieben hätten sein sollten, nicht fett waren. Dies wurde nach dem Test noch behoben.

10.2.3.3 Ergebnisse vom Systemtest 3 – 12.12.16 – Internet Explorer

Passed	Failed	Bemerkung
--------	--------	-----------

Test 1	X		Alle Logins mussten mit INTIGENAAD\ anfangen
Test 2	X		
Test 3	X		
Test 4	X		
Test 5	X		
Test 6	X		
Test 7	X		
Test 8	X		
Test 9	X		
Test 10	X		
Test 11	X		
Test 12	X		
Test 13	X		
Test 14	X		
Test 15	X		
Test 16	X		
Test 17	X		
Test 18	X		
Test 19	X		
Test 20	X		Zeiteingabe vorgefüllt mit 00:00:00
Test 21	X		
Test 22	X		
Test 23	X		
Test 24	X		
Test 25	X		
Test 26	X		
Test 27	X		
Test 28	X		
Test 29	X		
Test 30	X		
Test 31	X		
Test 32	X		
Test 33	X		
Test 34	X		
Test 35	X		
Test 36	X		
Test 37	X		
Test 38	X		
Test 39	X		
Test 40	X		
Test 41	X		
Test 42	X		
Test 43	X		
Test 44	X		

Tabelle 10: Ergebnisse vom Systemtest 3 - 12.12.16 - IE

Beim dritten Systemtest, der dieses Mal mit dem Internet Explorer durchgeführt wurde, konnte auch jeder Test erfolgreich durchgeführt werden. Jedoch verlangte der Internet Explorer die Eingabe der Domäne

des Active Directories (INTIGENAAD). Zusätzlich füllte der Internet Explorer alle Zeitfelder mit hh:mm:ss ab anstatt nur mit hh:mm, was aber zu keiner funktionalen Beeinträchtigung führte, da auch im Format hh:mm die Zeit eingegeben werden konnte.

10.3 Projektdiagramm

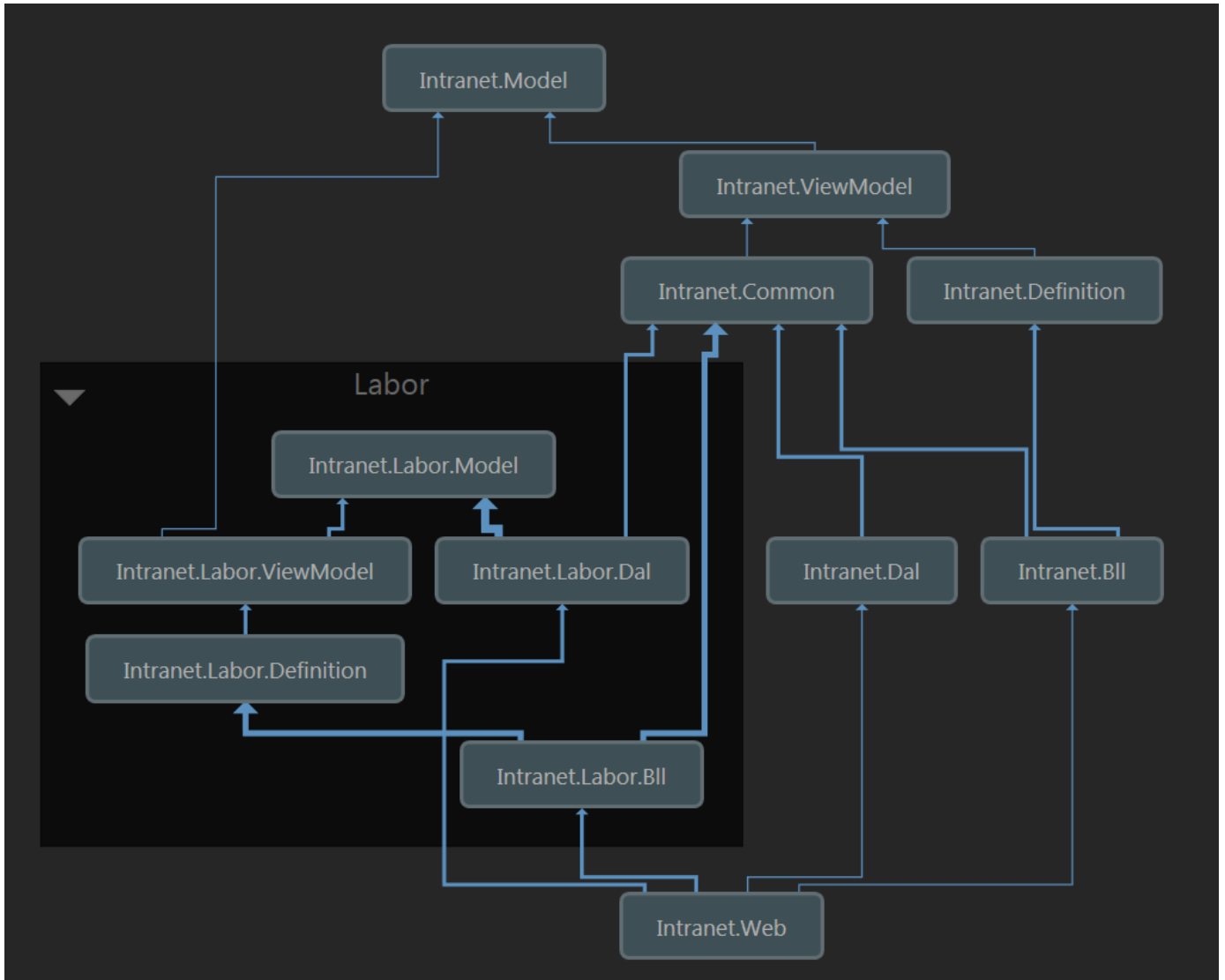


Abbildung 66: Dependency Diagramm Applikation

10.4 Package Diagramme

10.4.1 Package Diagramm Shell

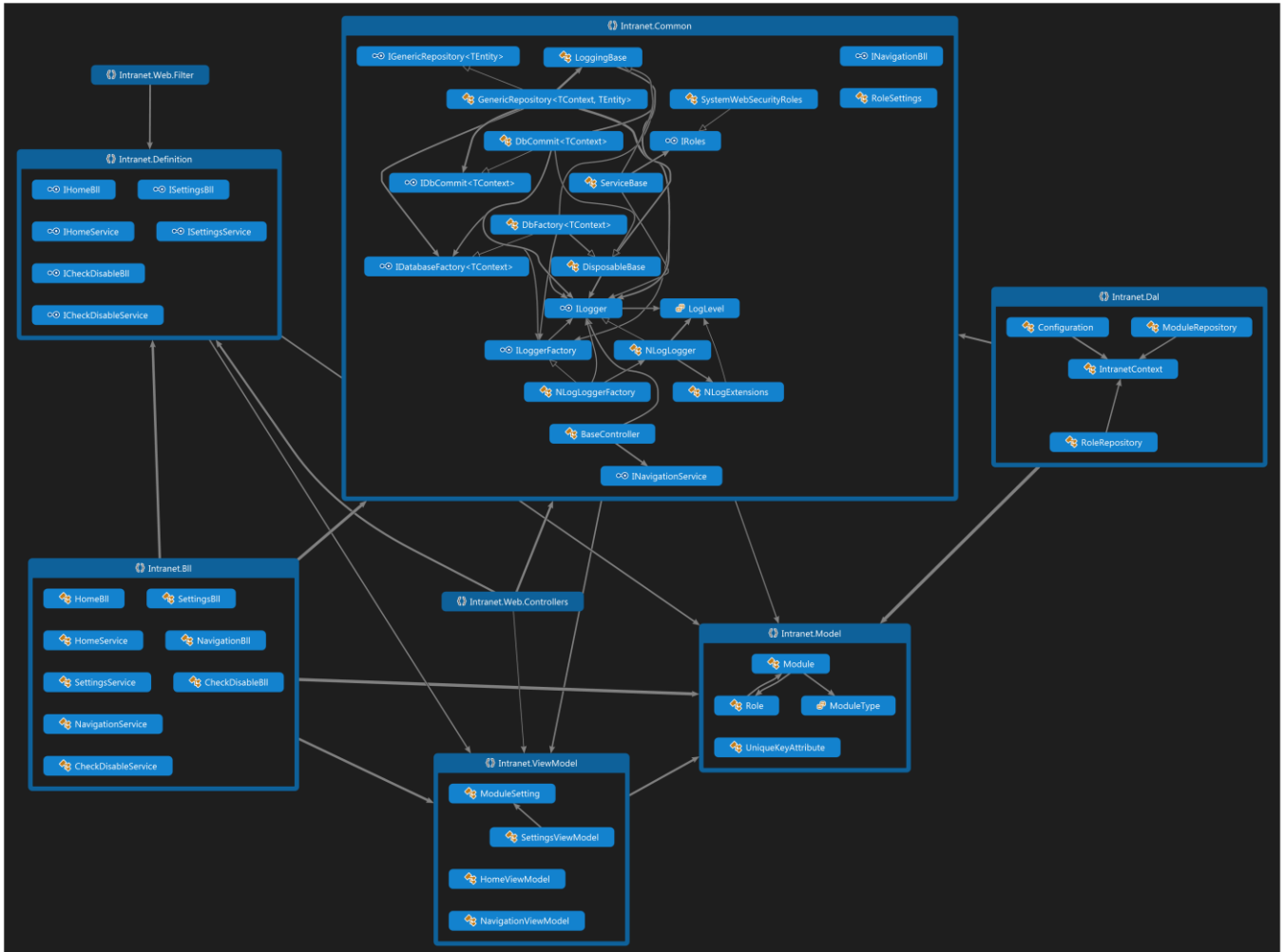


Abbildung 67: Package Diagramm Shell

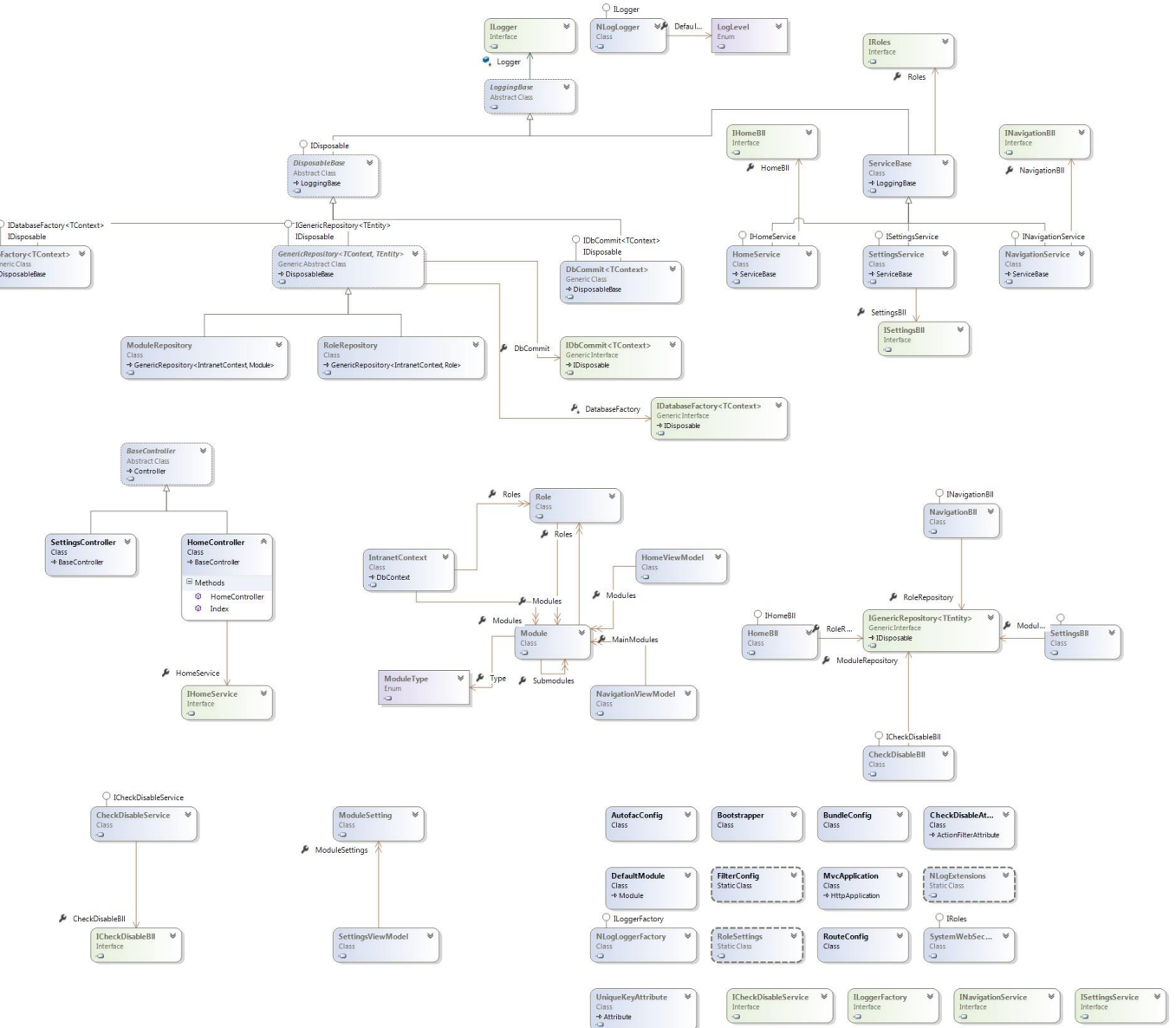
10.4.2 Package Diagramm Labor



Abbildung 68: Package Diagramm Labor

10.5 Klassen Diagramme

10.5.1 Intranet.X (Shell)



14

Abbildung 69: Klassendiagramm ganze Shell

¹⁴ Am besten als PDF ansehen und zoomen.

10.5.2 Intranet.Common

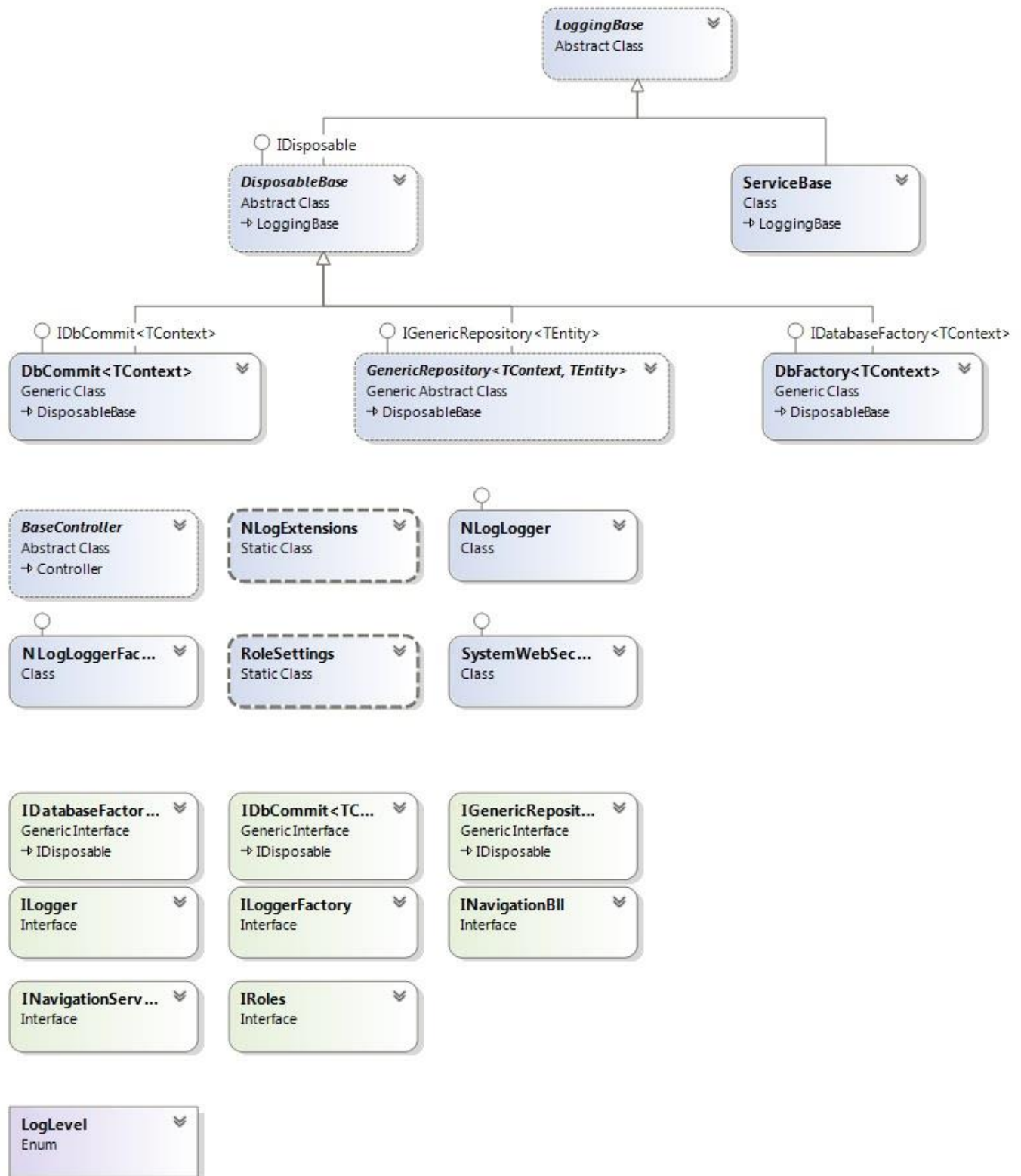


Abbildung 70: Klassendiagramm Common Übersicht

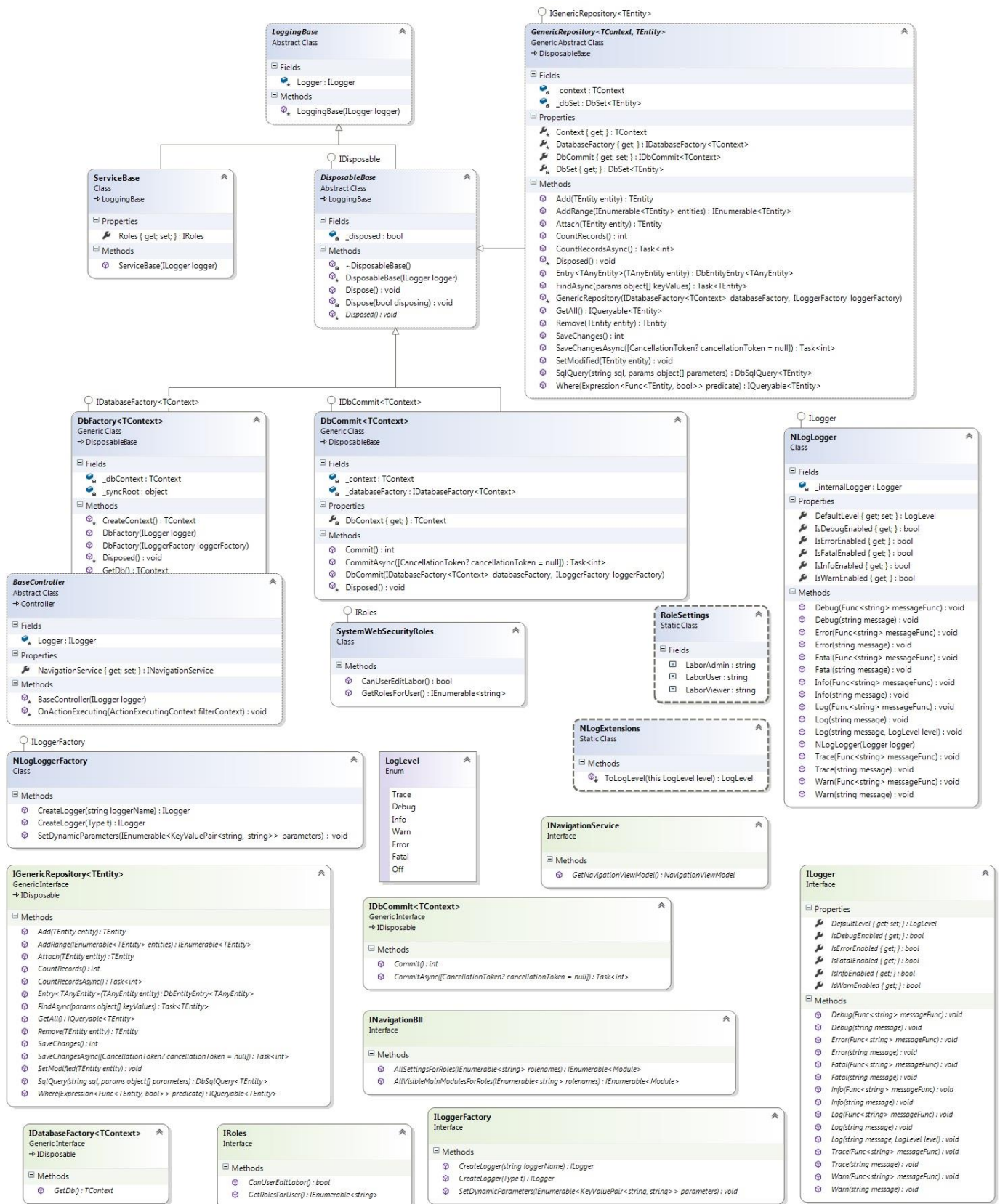


Abbildung 71: Klassendiagramm Common Detail

10.5.2.1 BII

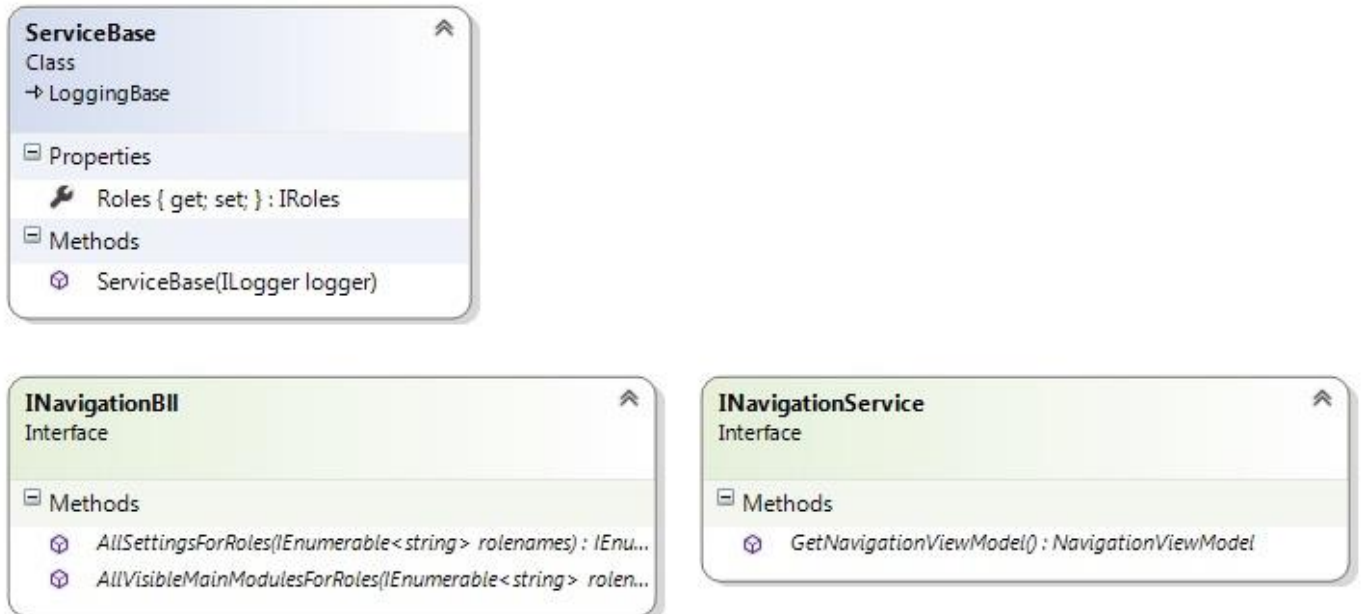


Abbildung 72: Klassendiagramm Intranet.Common BII

10.5.2.2 DB

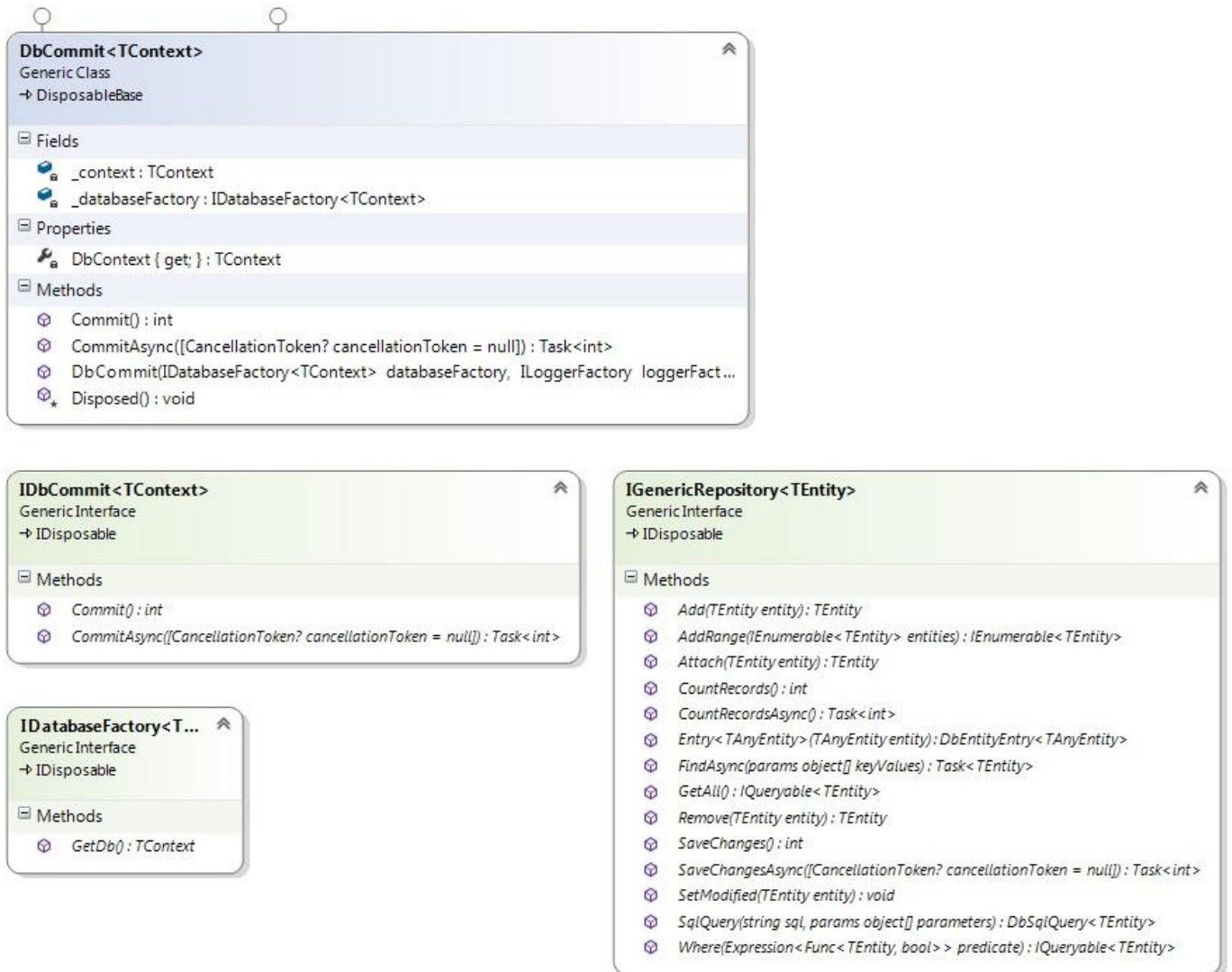


Abbildung 73: Klassendiagramm Intranet.Common Db

10.5.2.3 Disposable

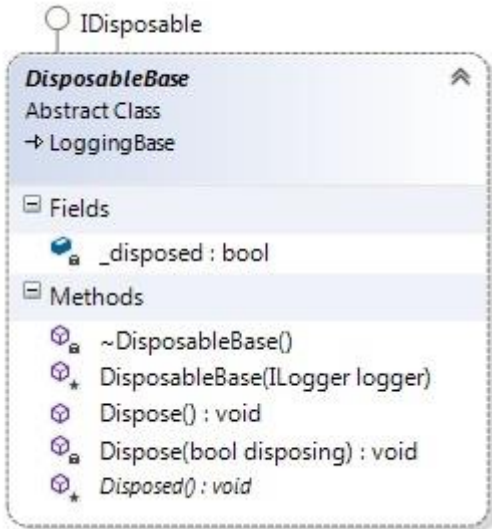


Abbildung 74: Klassendiagramm Intranet.Common Disposable

10.5.2.4 Logging

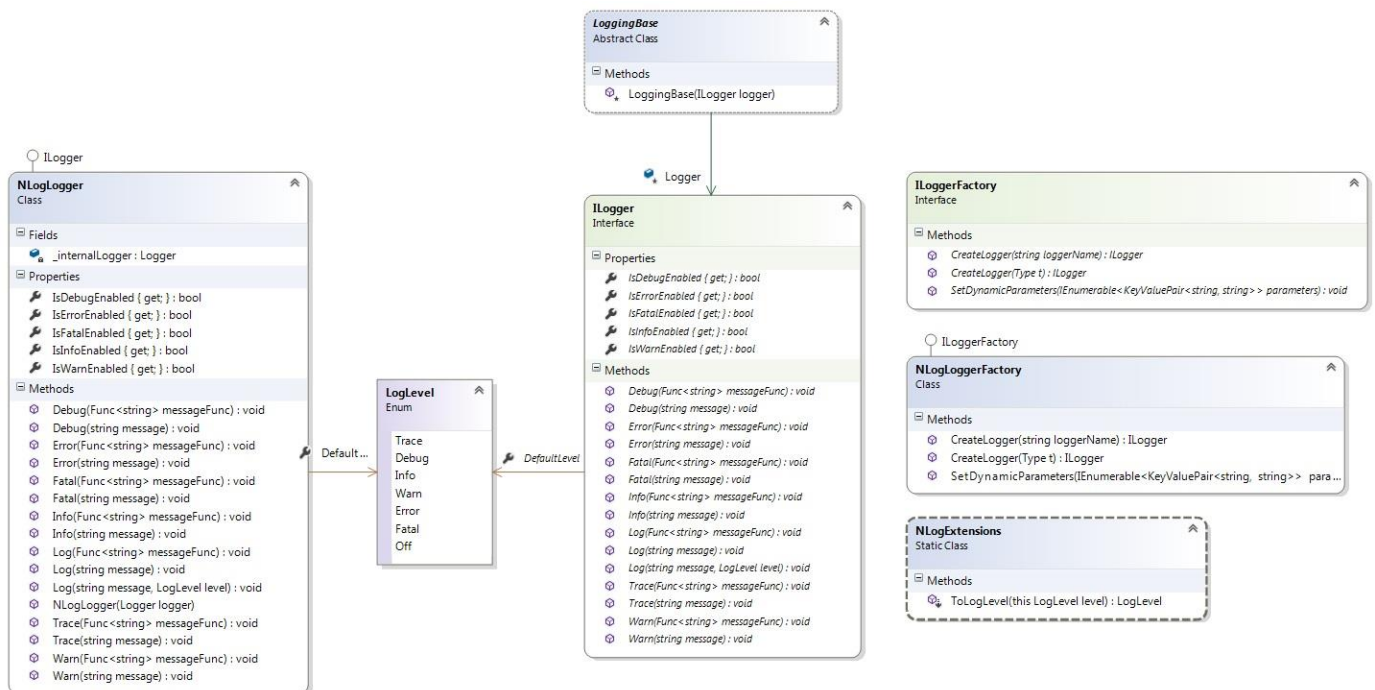


Abbildung 75: Klassendiagramm Intranet.Common Logging

10.5.2.5 Role



Abbildung 76: Klassendiagramm Intranet.Common Role

10.5.2.6 Web

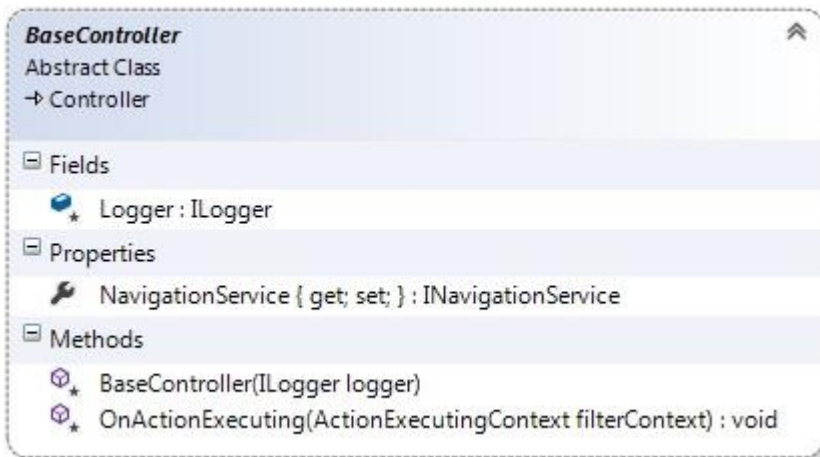


Abbildung 77: Klassendiagramm Intranet.Common Web

10.5.3 Intranet.Definition



Abbildung 78: Klassendiagramm Intranet.Definition

10.5.4 Intranet.BII



Abbildung 79: Klassendiagramm Intranet.BII Übersicht

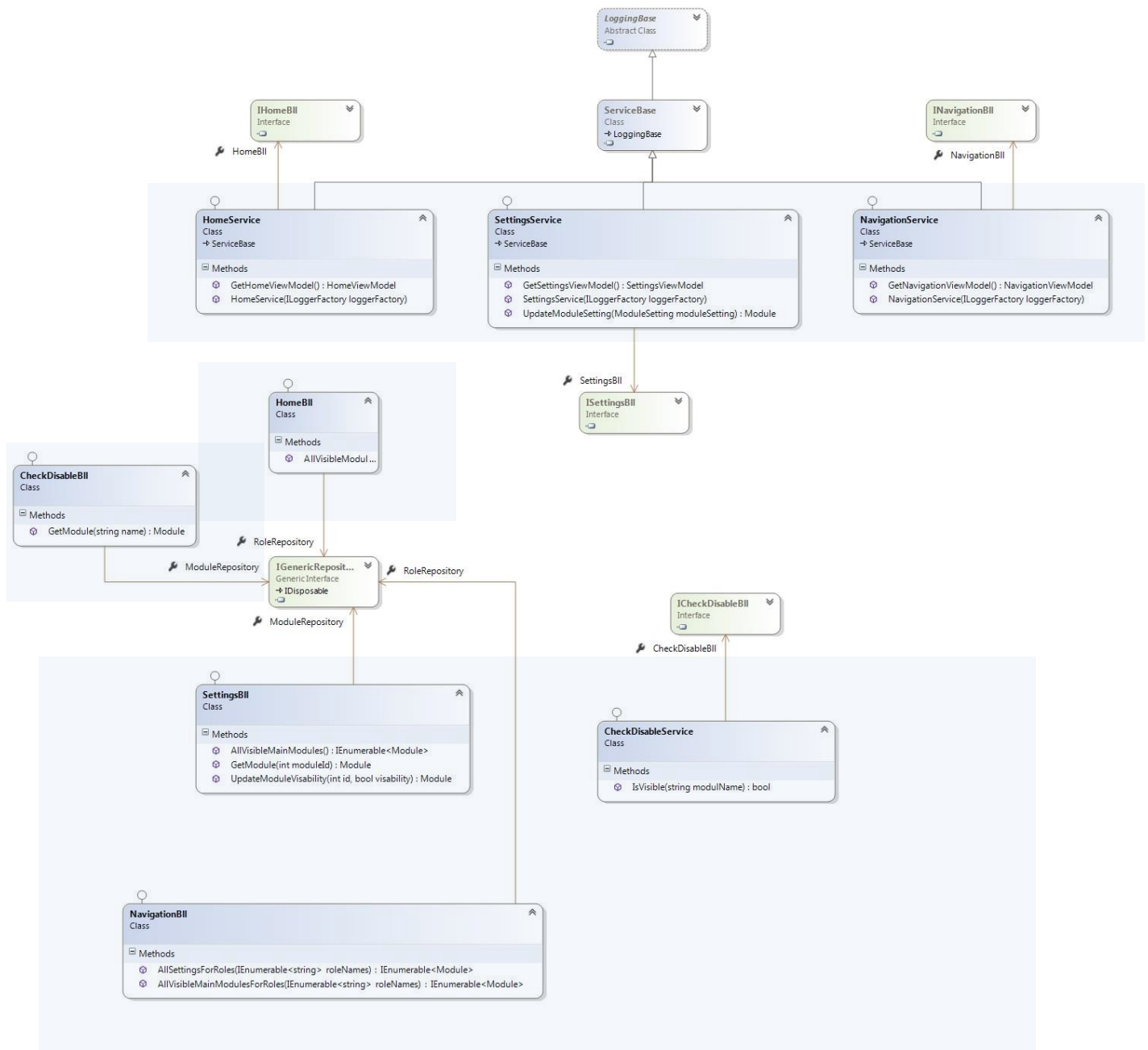


Abbildung 80: Klassendiagramm Intranet.BII Detailliert

10.5.5 Intranet.Dal

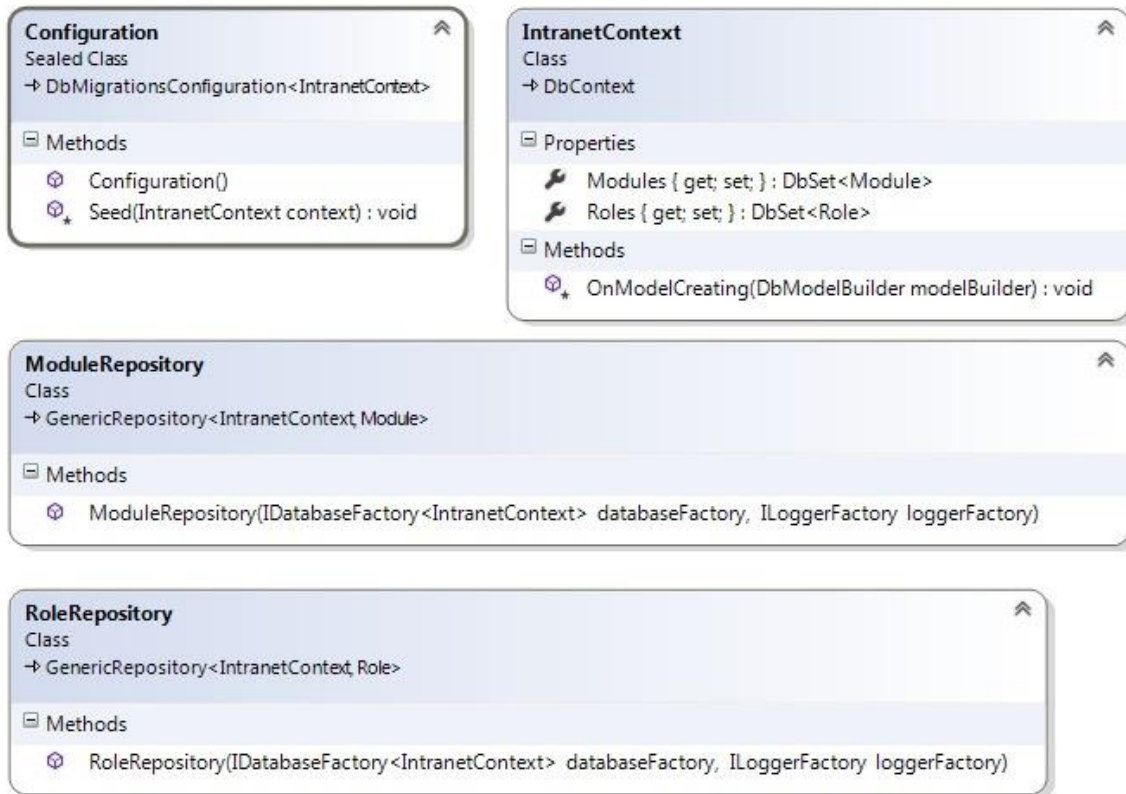


Abbildung 81: Klassendiagramm Intranet.Dal

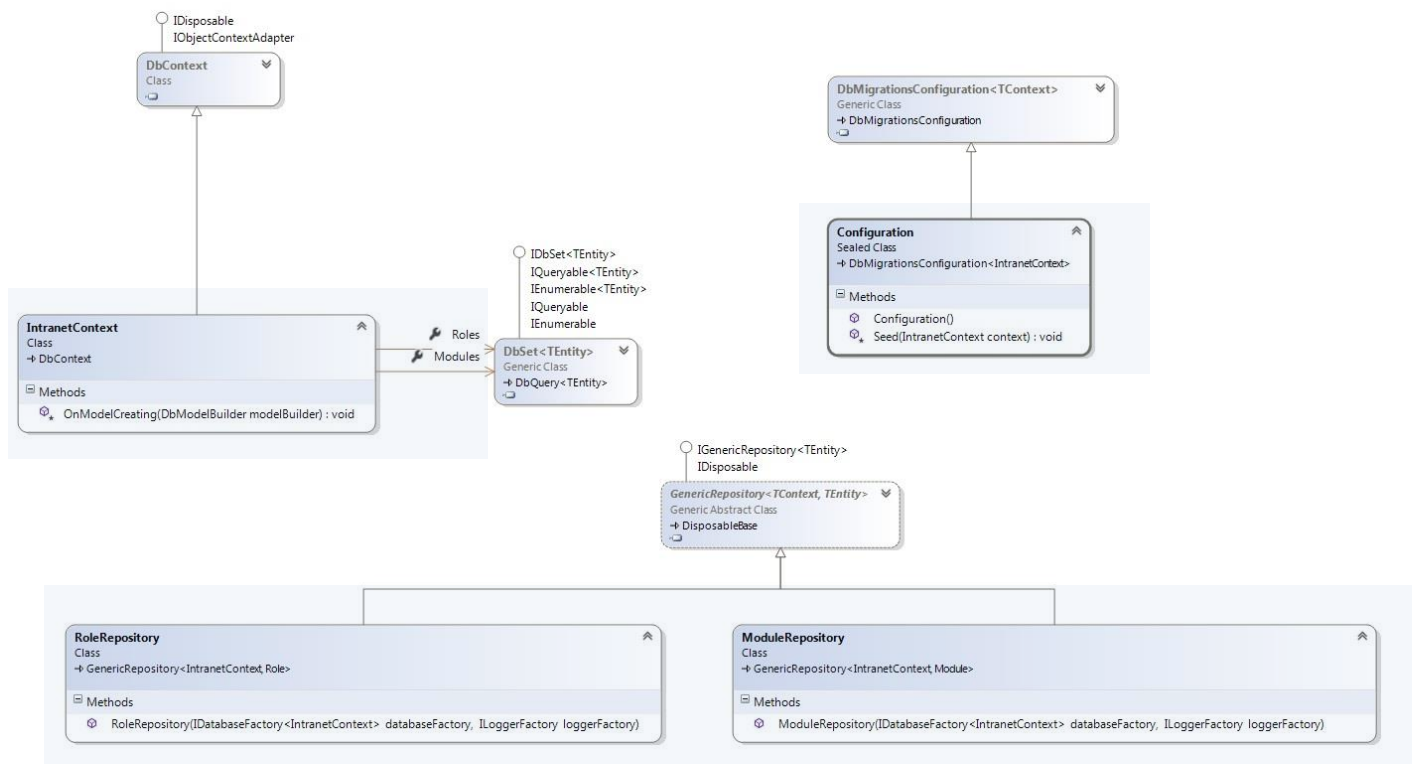


Abbildung 82: Klassendiagramm Intranet.Dal detailliert

10.5.6 Intranet.Model

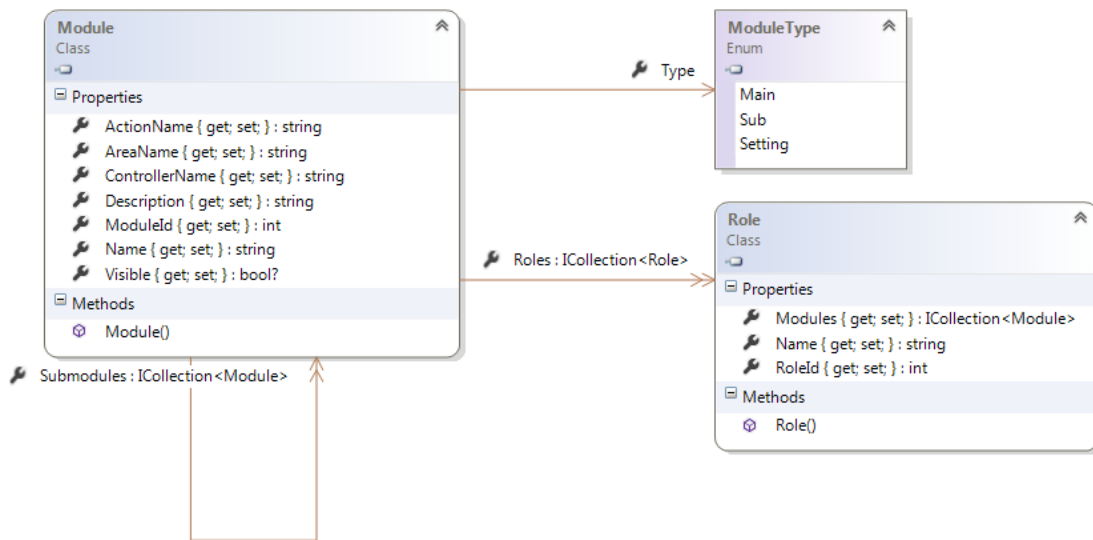


Abbildung 83: Klassendiagramm Model Shell

10.5.7 Intranet.ViewModel

Unser ViewModel für die Shell sieht wie folgt aus.

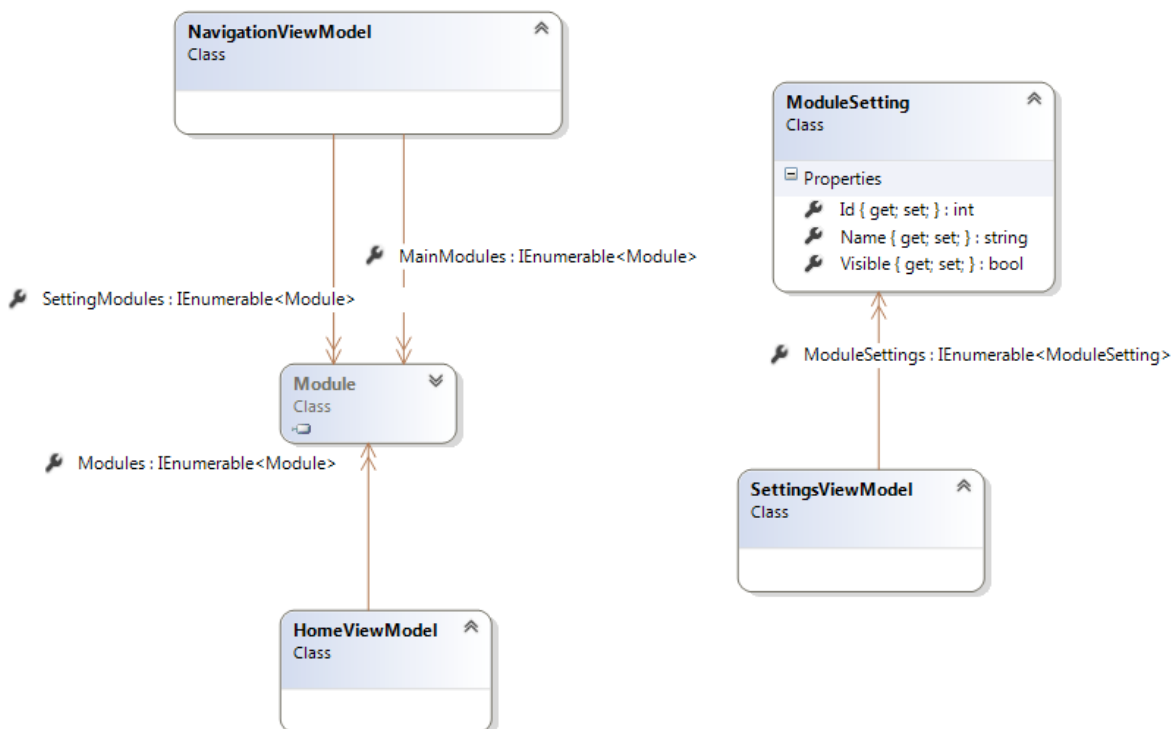


Abbildung 84: Klassendiagramm ViewModel Shell

HomeViewModel ist das ViewModel für Home, die Startseite von Intranet

Modules	ist eine Liste von Module welche in der View Home dann angezeigt wird.
SettingsViewModel	ist das ViewModel für Setting, die Seite in welche man Module ein und ausblenden kann.
ModuleSettings	ist eine Liste von ModuleSetting welche die Module zum ein und ausblenden darstellt
ModuleSetting	ist eine Klasse welche Module abbildet und nur die Daten beinhaltet welche das SettingsViewModel braucht
Id	ist die Id des Modules
Name	Ist der Modul Name welche in der View dargestellt wird.
Visible	Ist der Boolean welcher sagt, ob das Modul angezeigt wird, oder ausgeblendet.
NavigationViewModel	ist das ViewModel für die gesamte Navigation.
MainModules	ist eine Liste von Modulen mit dem Typ Main.
SettingModules	ist eine Liste von Modulen mit dem Typ Sub

Tabelle 11: ViewModel Shell Werte Beschreibung

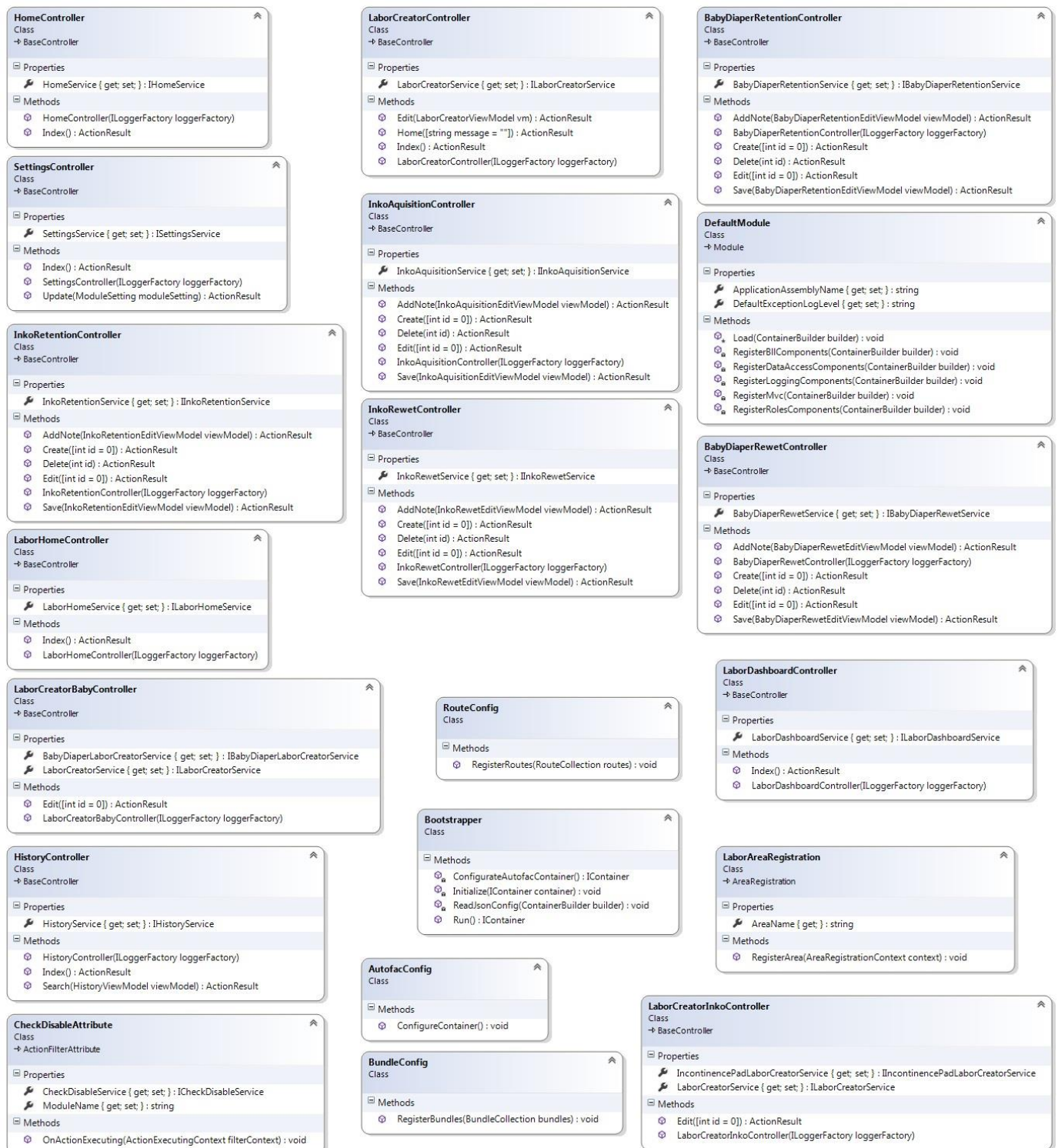


Abbildung 86: Klassendiagramm Intranet.Web Detail

10.5.9 Intranet.Labor.Model

10.5.9.1 Intigena Stammdaten

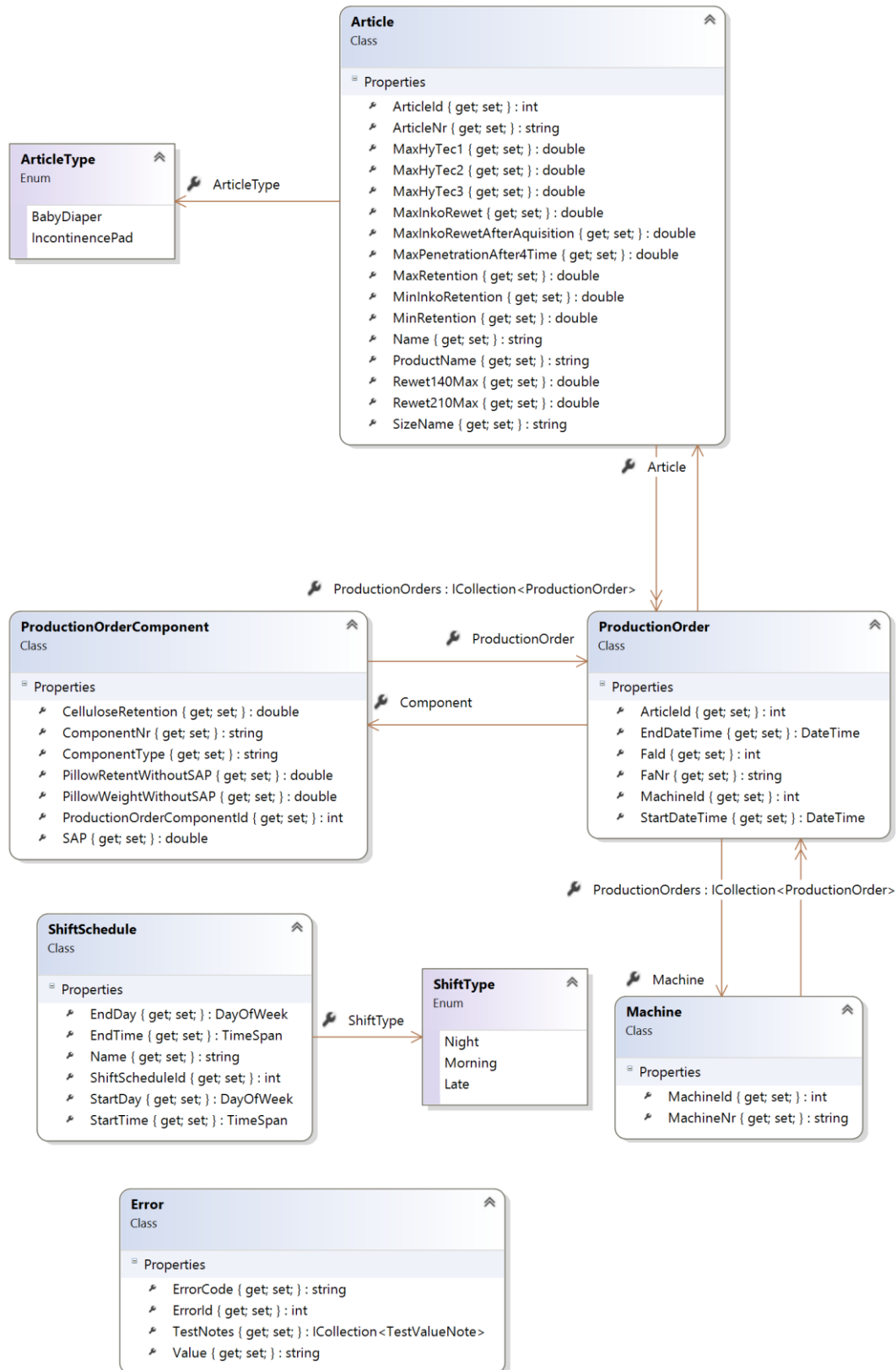


Abbildung 87: Model Fertigungsartikel

10.5.9.2 Labor

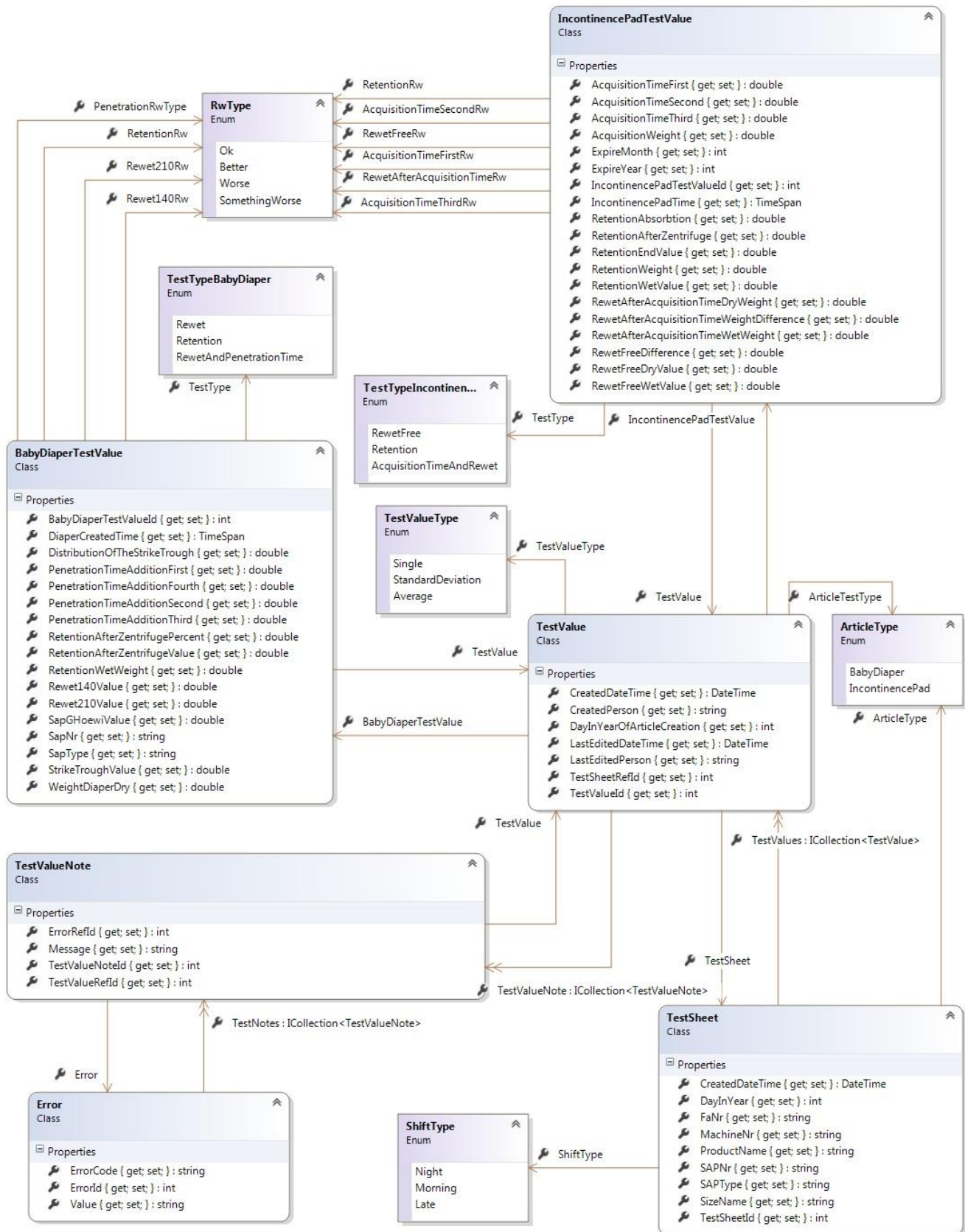


Abbildung 88 Model Labordaten

10.5.9.3 Fertigungsartikel

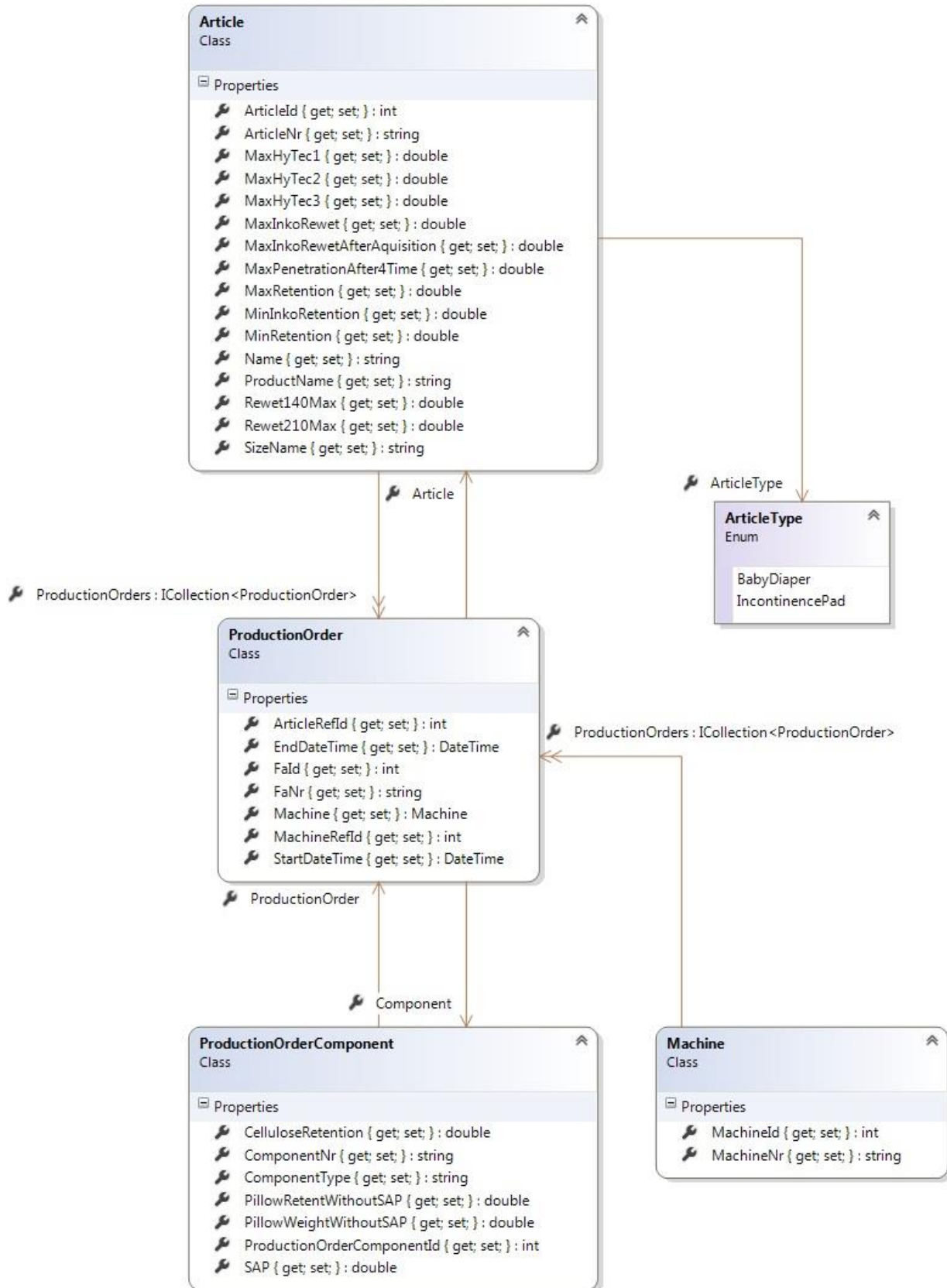


Abbildung 89: Model Fertigungsartikel

10.5.9.4 Instron



Abbildung 90: Viewmodel Instron

10.5.9.5 Schichtplan

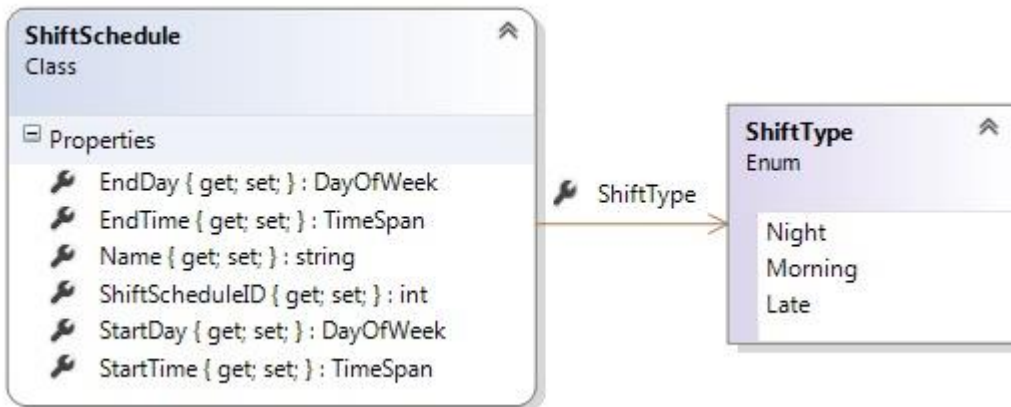


Abbildung 91: Model Schichtplan

10.5.10 Intranet.Labor.BII



Abbildung 92: Klassendiagramm Labor.BII Übersicht

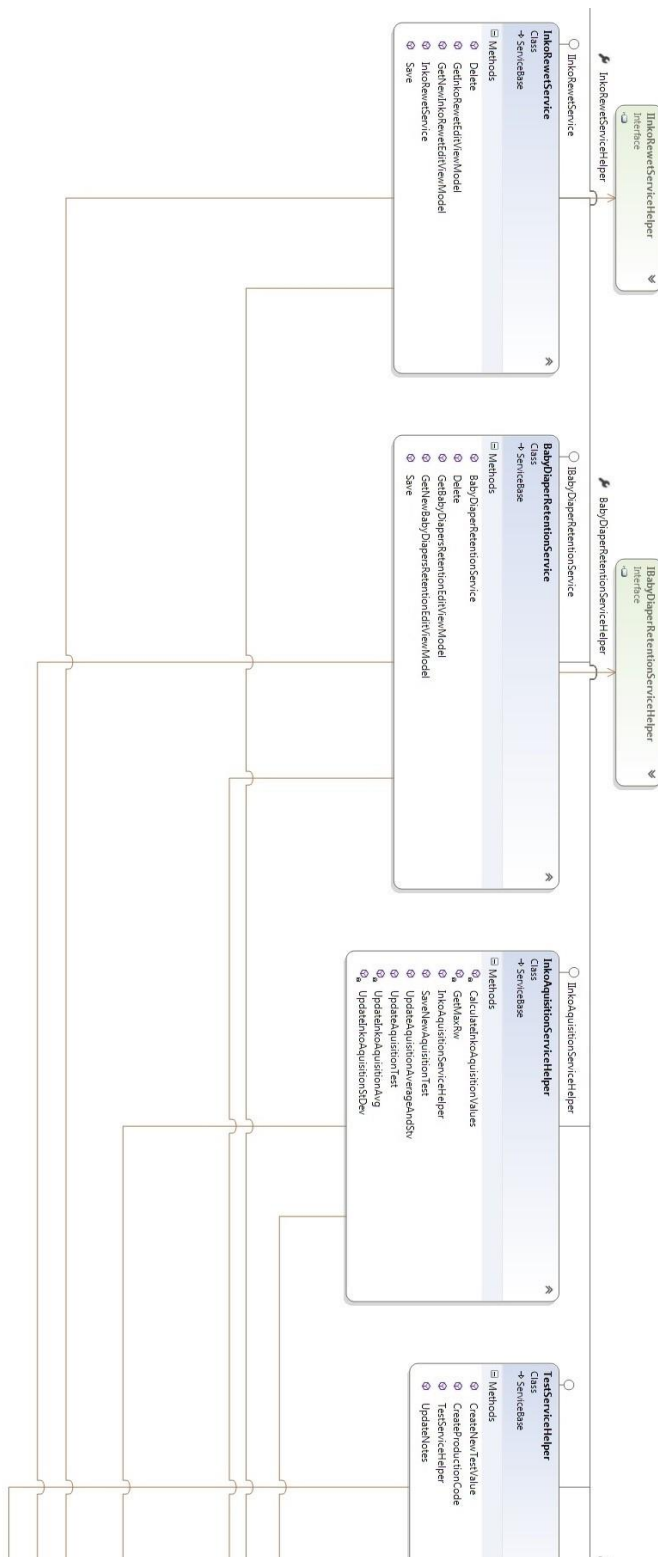


Abbildung 94: Klassendiagramm Labor.BII Detail Teil 2

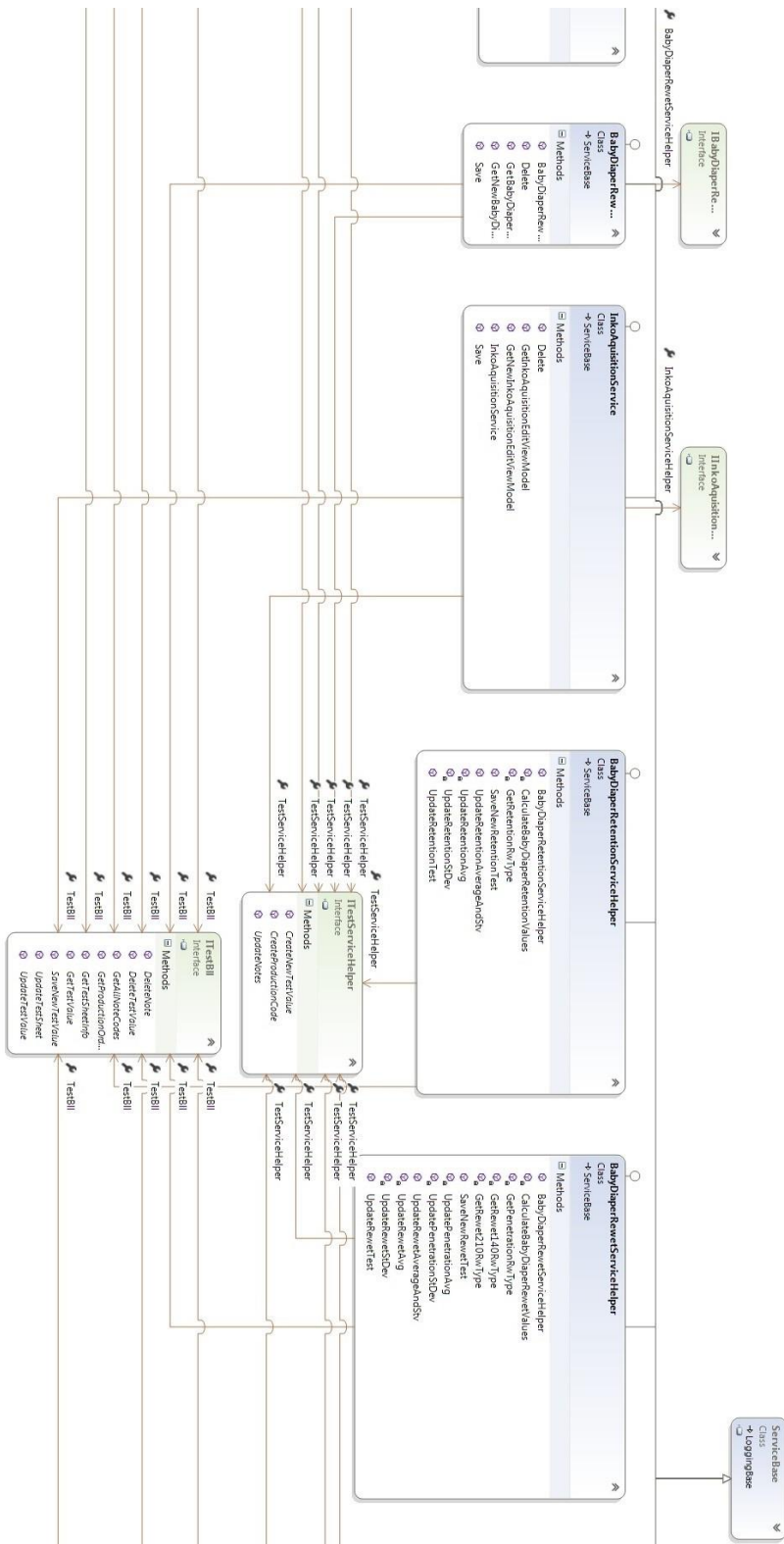


Abbildung 95: Klassendiagramm Labor.Bll Detail Teil 3

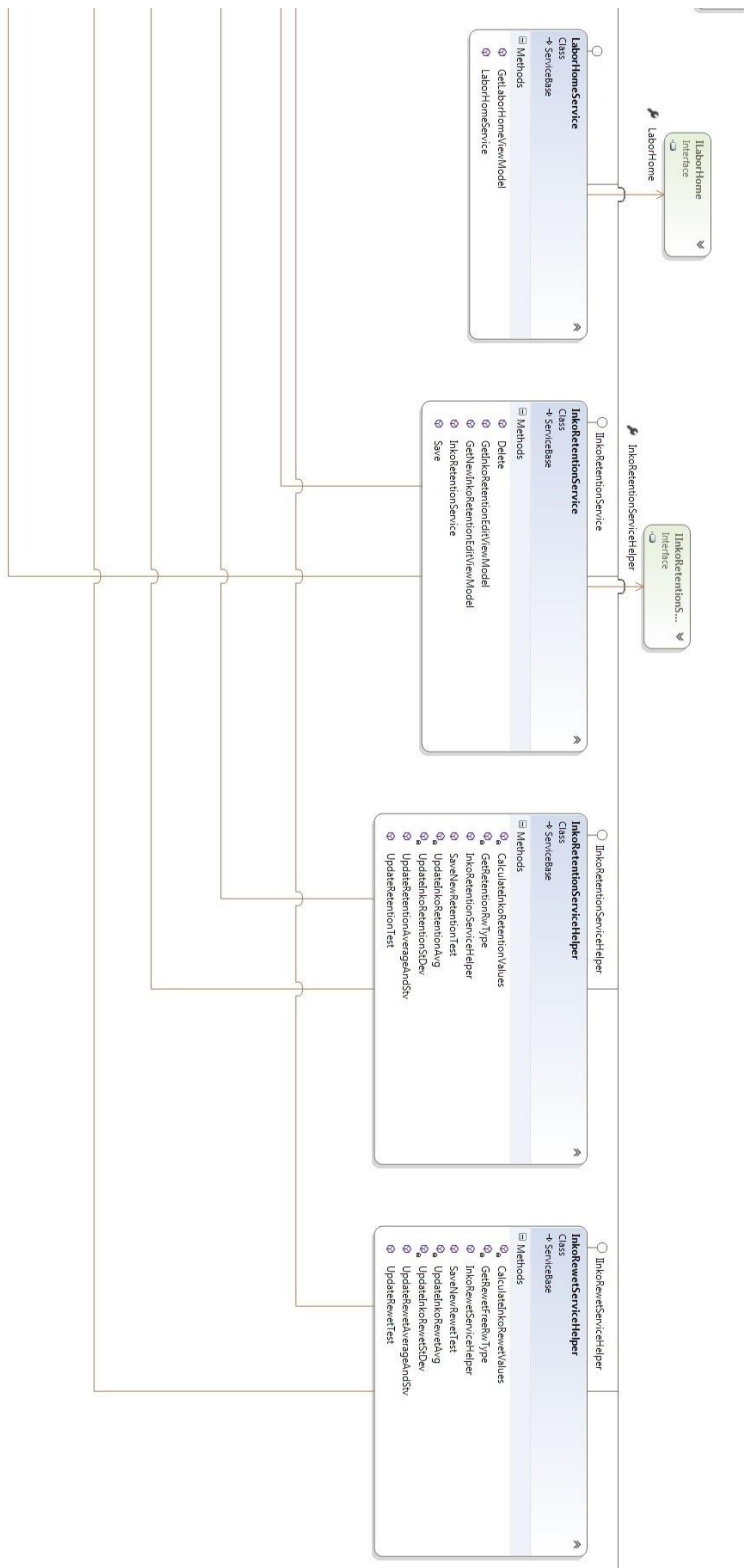


Abbildung 96: Klassendiagramm Labor.BII Detail Teil 4

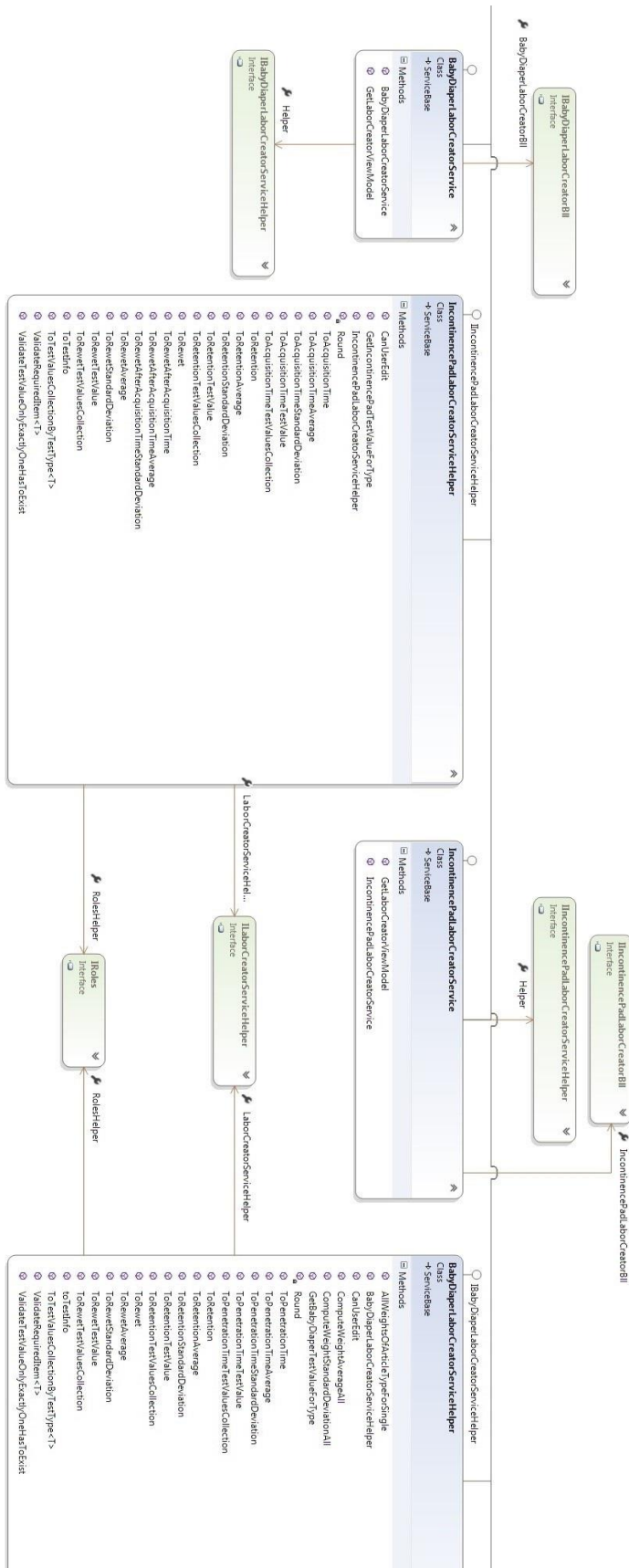


Abbildung 97: Klassendiagramm Labor.Bll Detail Teil 5

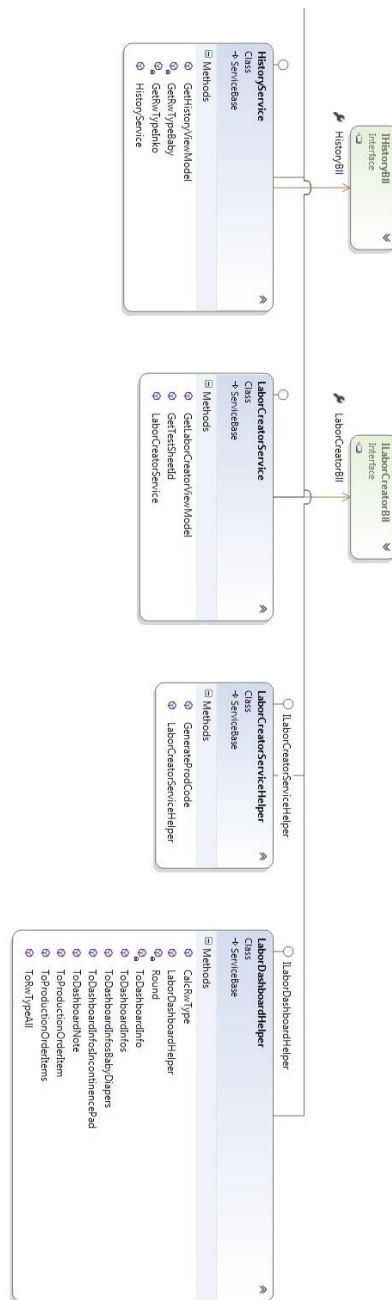


Abbildung 98: Klassendiagramm Labor.Bll Detail Teil 6

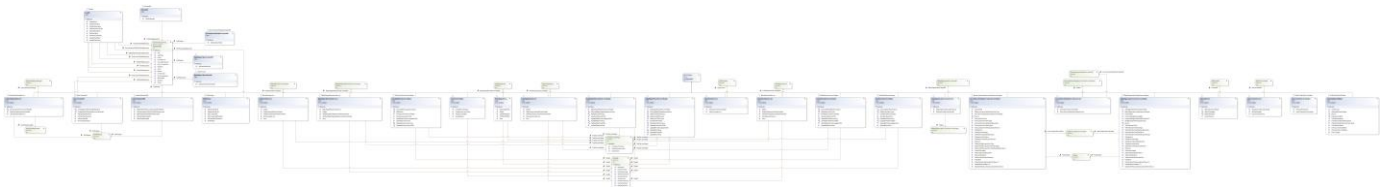


Abbildung 99: Klassendiagramm Labor.BII Detail gesamt

10.5.11 Intranet.Labor.Dal

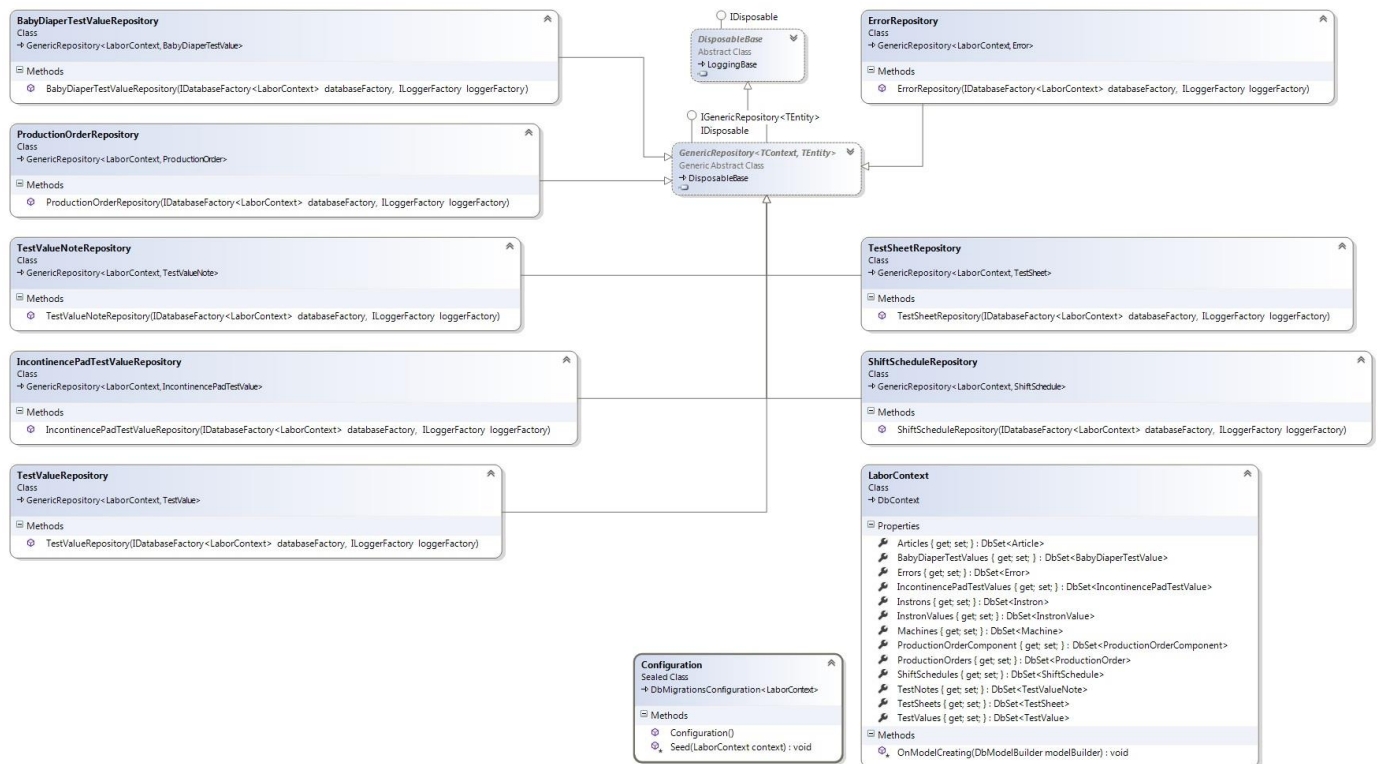


Abbildung 100: Klassendiagramm Labor.Dal

10.5.12 Intranet.Labor.Definition

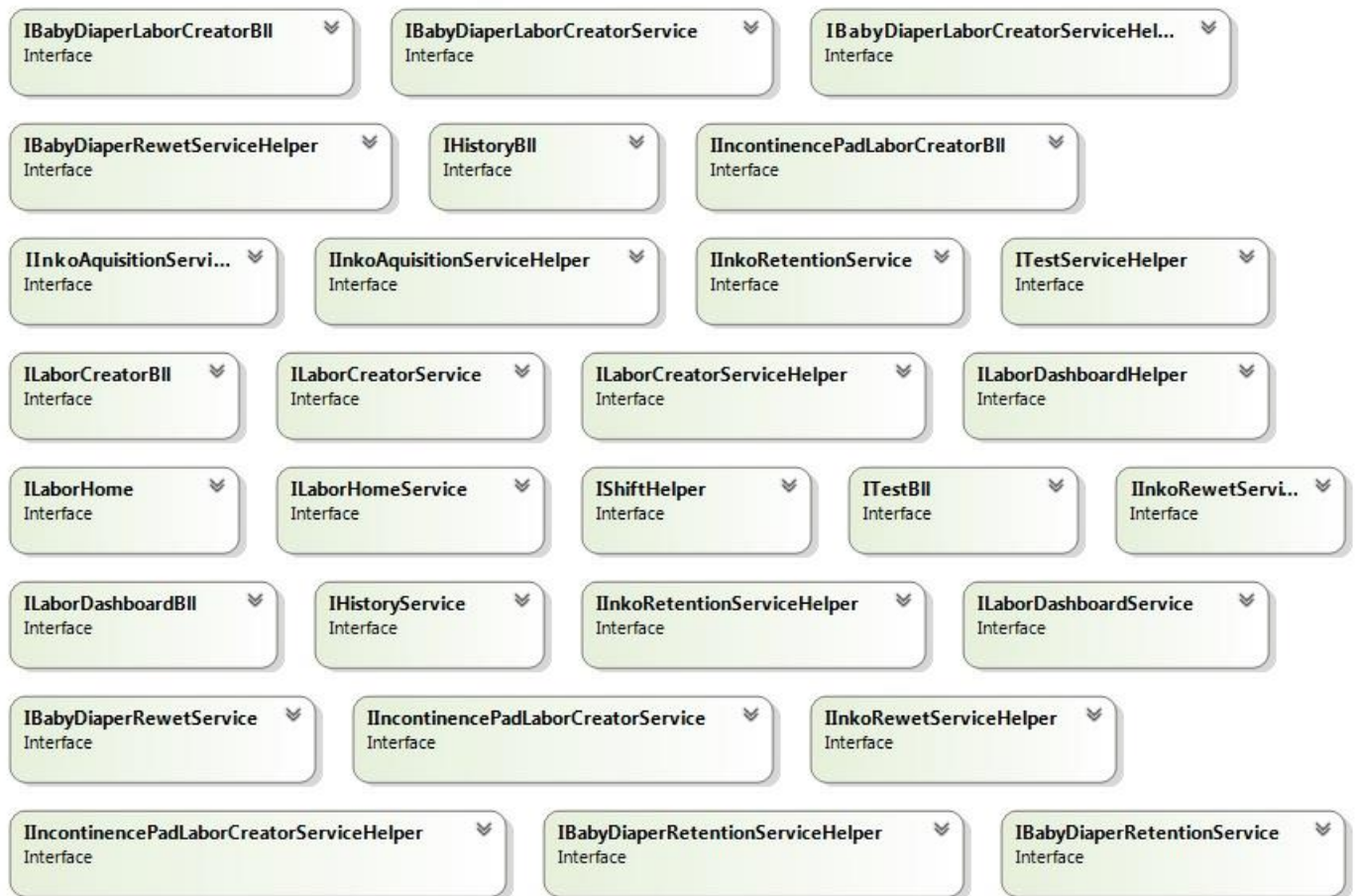


Abbildung 101: Klassendiagramm Labor.Definition Übersicht

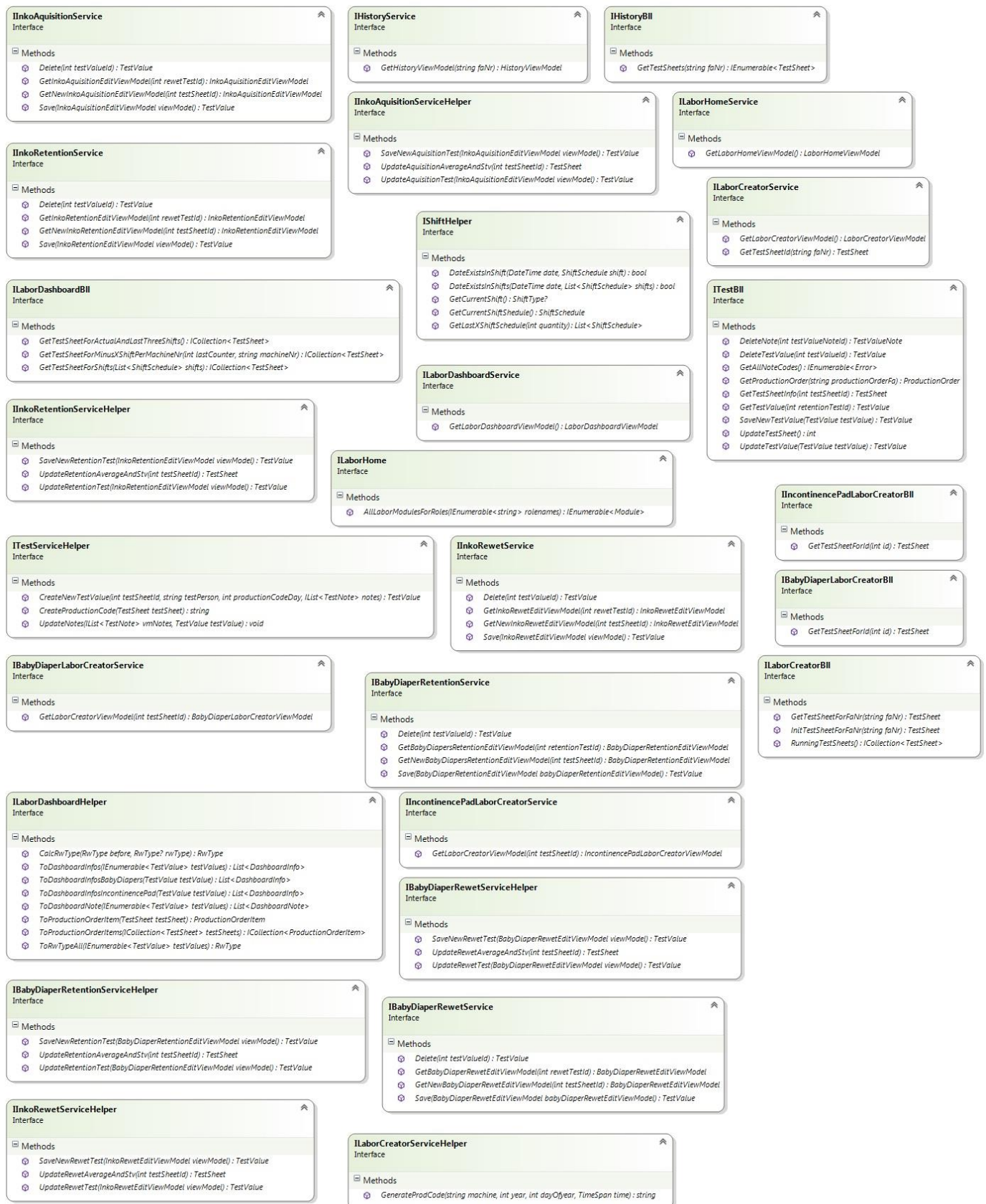


Abbildung 102: Klassendiagramm Labor.Definition Detail Teil 1

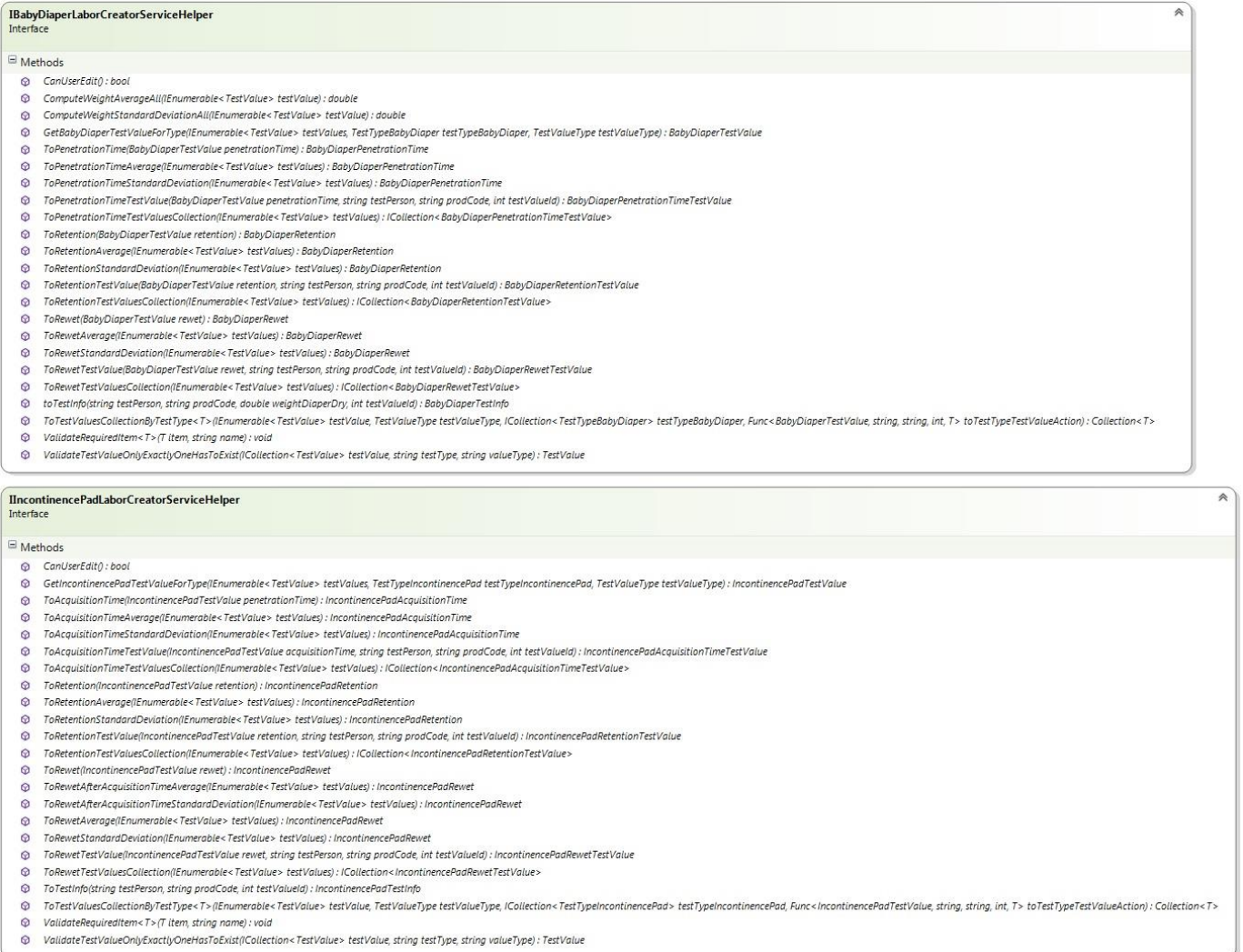


Abbildung 103: Klassendiagramm Labor.Definition Detail Teil 2

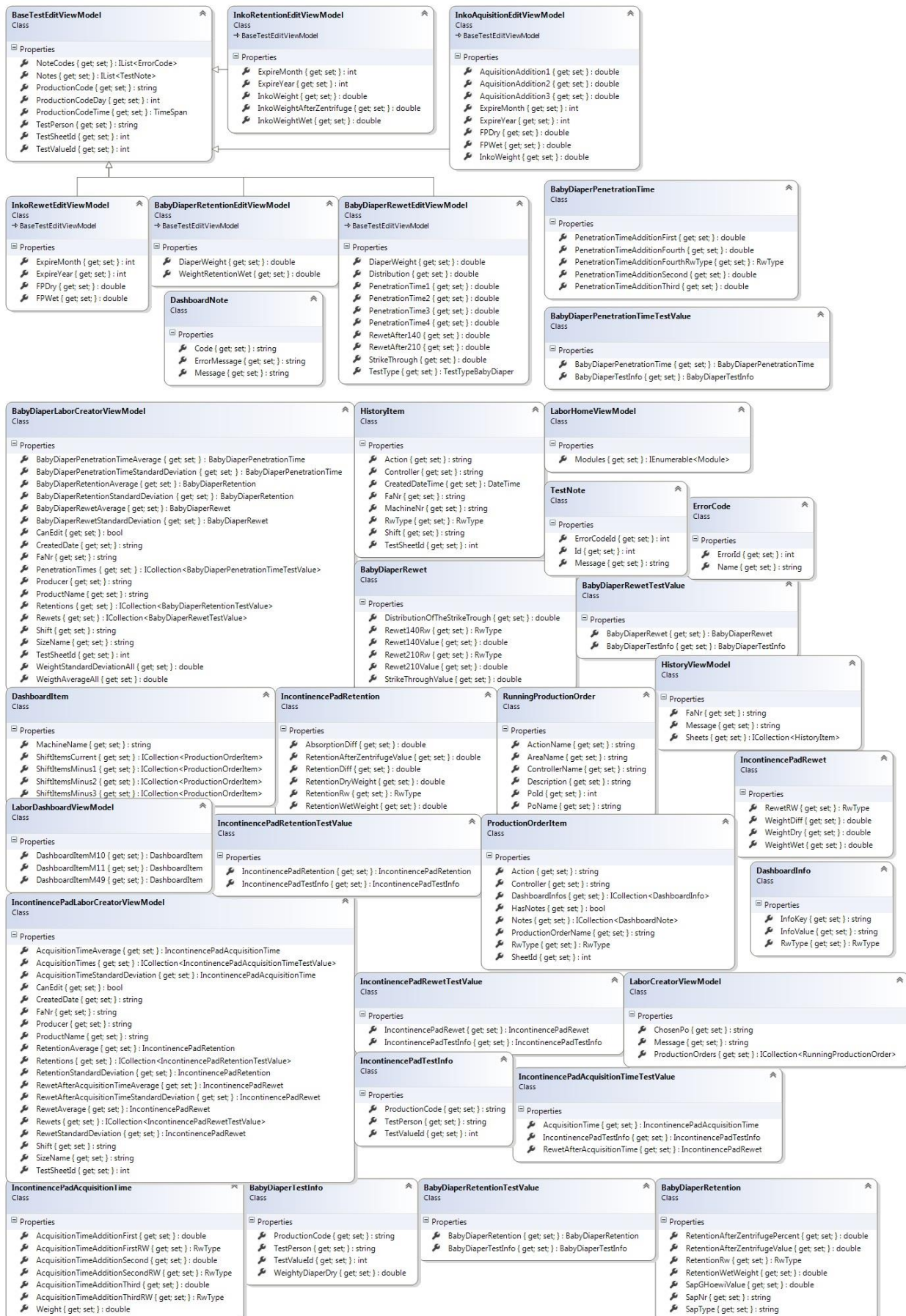


Abbildung 105: Klassendiagramm Labor.ViewModel Detail

10.6 Separate Dokumente

- Projektantrag
- Projektplan (Alt)
- Projektplan
- Anforderungsspezifikation Shell
- Anforderungsspezifikation Modul Labor, QS
- Anforderungsspezifikation XML
- Benutzerhandbuch Labor
- Sitzungsprotokolle
 - 21.09.2016
 - 28.09.2016
 - 03.10.2016
 - 11.10.2016
 - 18.10.2016
 - 25.10.2016
 - 04.11.2016
 - 17.11.2016
 - 30.11.2016
 - 06.12.2016
 - 13.12.2016
 - 20.12.2016
- Sourcecode verfügbar unter: <https://github.com/ClaudiaSaxer/Intranet>
- API Beschreibung verfügbar unter: <https://claudiasaxer.github.io/Intranet/>