Projekt Stundenplan

Inhaltsverzeichnis

Inhaltsverzeichnis

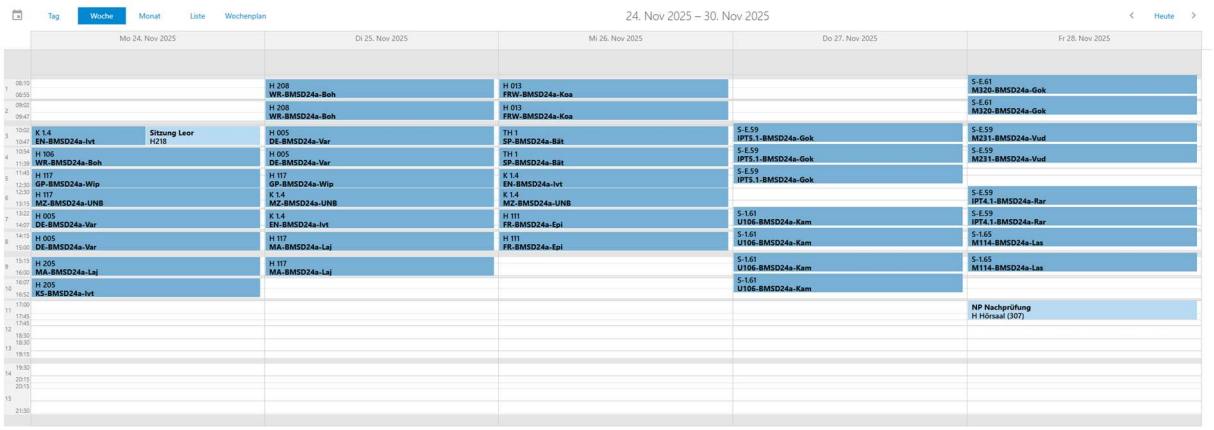
Inhaltsverzeichnis 1
Projektbeschreibung 2
Anforderungen & Usecases 3
Analyse & Planung 4
Umsetzung 6
Bedienungsanleitung 7
Qualitätskriterien & Bewertung 9
Tests & Beispielsausgabe 12
Arbeitsrapport & Zeitaufwand 15

Abzugeben bis 27.11.2025 23:59

Projektbeschreibung

Unser Projekt hat zum Ziel, einen dynamischen Stundenplan-Manager zu entwickeln, der für jede anfragende Person, sei es eine Schülerin, ein Schüler oder eine Lehrperson einen exakten und personalisierten Plan zurückgibt. Die zentrale Stundenplan-Übersicht ist dabei auf maximale Klarheit ausgelegt und liefert alle notwendigen Informationen: Neben den genauen Zeiten werden das unterrichtete Fach, die zuständige Lehrperson, der spezifische Raum sowie die entsprechende Klasse angezeigt. Ein besonderer Fokus liegt auf der visuellen Darstellung; unser Anspruch ist es, eine ästhetisch ansprechende und intuitive Benutzeroberfläche zu schaffen, deren Übersichtlichkeit sich an den bewährten Standards des Schulnetz-Stundenplans orientiert. Ergänzend zur reinen Anzeigefunktion bietet das System erweiterte Verwaltungsmöglichkeiten, welche es Nutzern erlauben, neue Entitäten hinzuzufügen. Dies umfasst die Möglichkeit, neue Schüler, Räume, Lehrpersonen oder Fächer in die Json Datei aufzunehmen, um die Aktualität und Vollständigkeit des Stundenplans jederzeit zu garantieren.

So ähnlich wie das Schulnetz

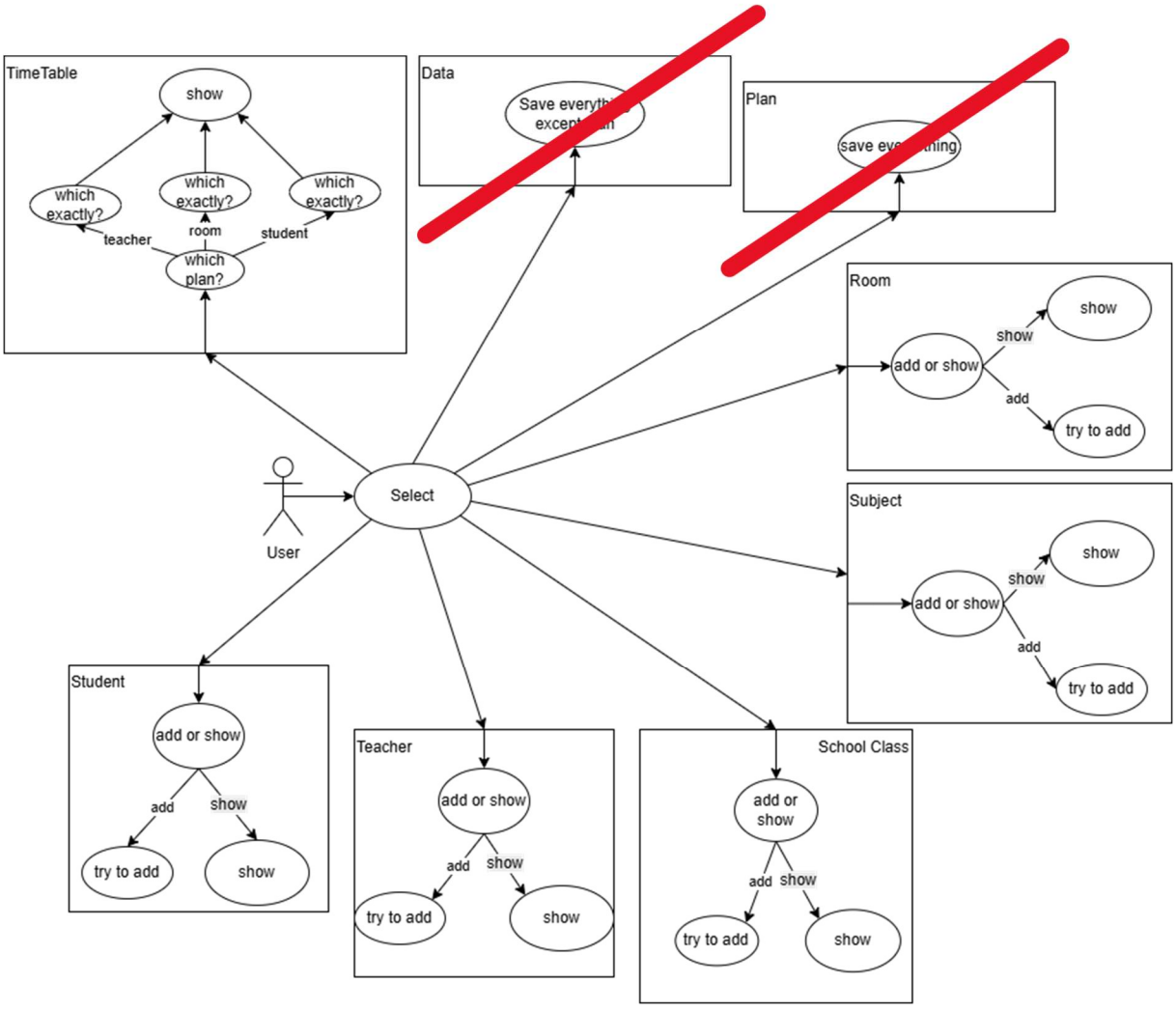


Anforderungen & Usecases

Die Anforderung war einen Automatischer Stundenplan Generator zu haben, mit welchem man die Verfügbarkeit der Räume, Lehrpersonen und Schüler zeigen kann. Es sollte ebenfalls möglichst wenig Räume parallel gebraucht werden. Es sollte ebenfalls möglich sein nach einem Neustart immer noch Zugang zu Generierten Plänen zu haben. Es wird mehr Wichtigkeit daraufgelegt, dass die Räume und Lehrpersonen nicht gleichzeitig genutzt werden als die Randzeiten und die Zwischenstunden, welche nur auf die „Ästhetik“ eines Plans, Einfluss haben.

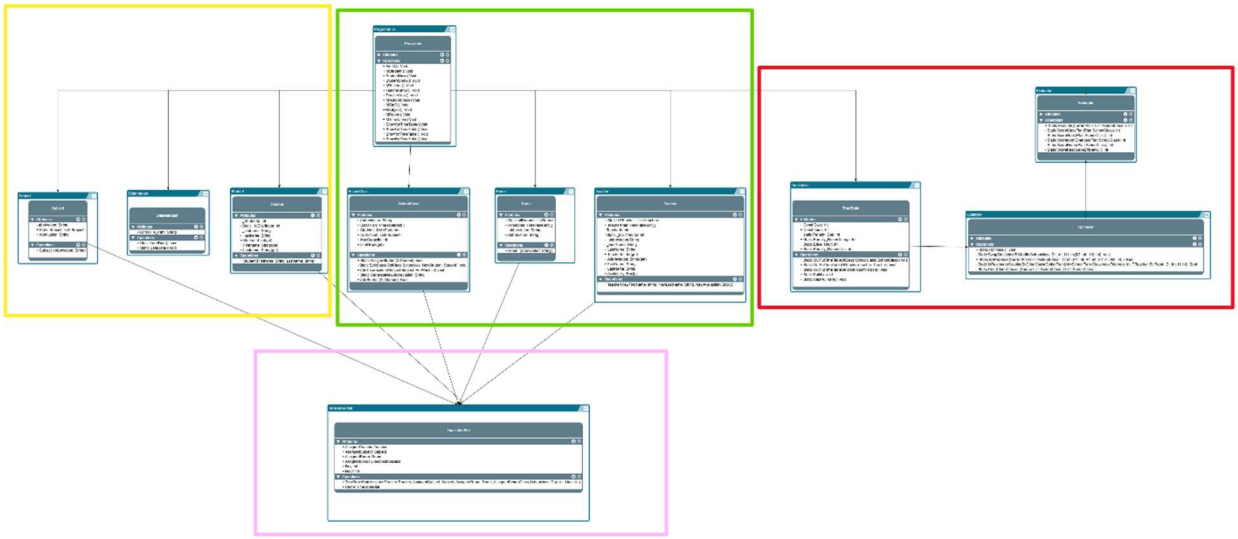
Das Ergebnis, welches der Kunde erhalten möchte, ist ein sich selbst erstellender gut optimierter Stundenplan, welcher sich auf die Anzahl Räume sowie die Arbeitszeiten der Lehrer beachtet. Noch dazu sollte es Beachten, dass dir Räume Frei sind und die Schulzeiten gut aufgeteilt sind und auch die Randstunden weniger gesetzt ist.

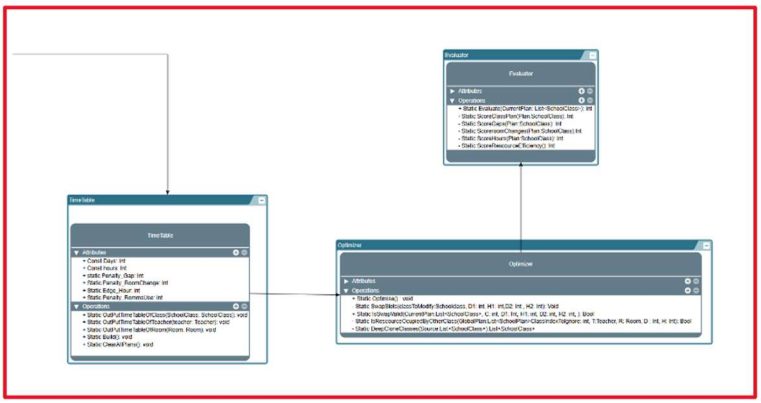
Use Case Diagramm:
Rot wurde Markiert was in Verlauf des Projektes verworfen wurde.

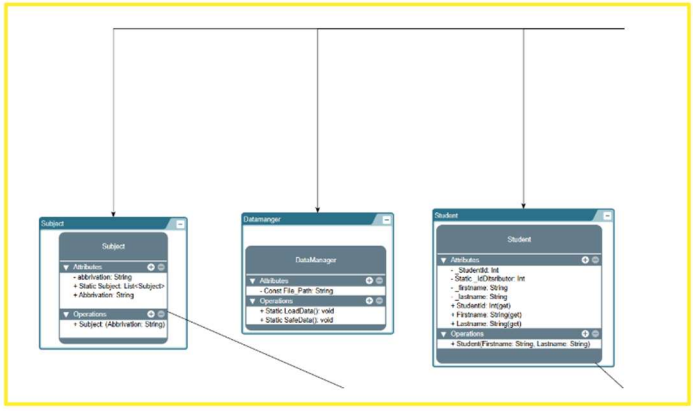


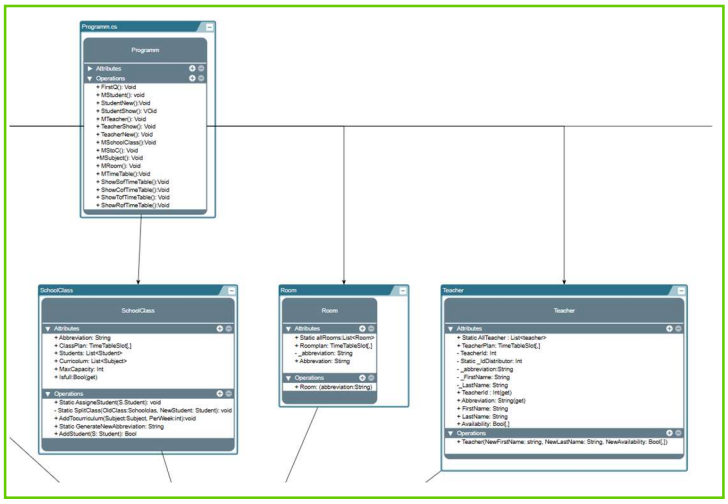
Analyse & Planung

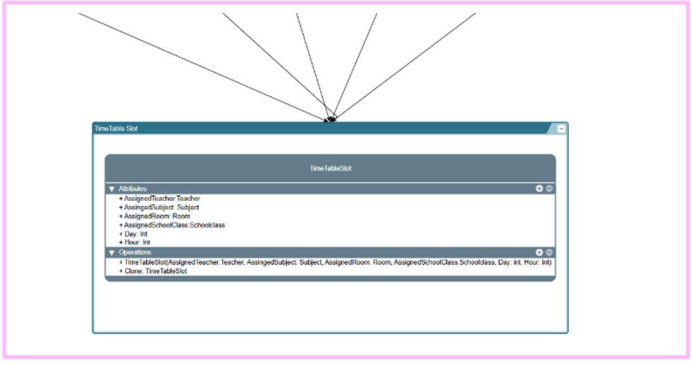
Bei uns war es wichtig, dass wir alles in eigene Klassen oder, wenn möglich, in eigene Methoden auslagern, um einen guten Überblick über den Code zu behalten. Beim Algorithmus fiel unsere Entscheidung auf drei Klassen, durch welche die ganze Stundenplan-Generation funktioniert. Timetable steuert die Darstellung der Stundenpläne und ist zuständig dafür, dass der erste Plan korrekt, aber vielleicht nicht perfekt, generiert wurde. Optimizer verbessert – wie schon der Name sagt – die Pläne, die generiert wurden, und der Evaluator bewertet die einzelnen Stundenpläne anhand der Kriterien, die durch unseren Kunden gesetzt wurden. Für eine persistente Speicherung entschieden wir uns für ein JSON-File. Dieses File beinhaltet alle Klassen und deren Werte und auch alle Pläne, die generiert wurden und die in einem 2D-Array gespeichert wurden, um den Speicherplatz zu minimieren und den sicheren Zugriff auf die richtige Stunde am richtigen Tag zu gewährleisten.




Evaluator
Evaluator
TimeTable
TimeTable
Optimizer
Optimizer


Subject
Subject
Datamanger
DataManager
Student
Student


Programm.cs
Programm
SchoolClass
SchoolClass
Room
Room
Teacher
Teacher


TimeTable Slot
TimeTableSlot

Umsetzung

OutputTimetableofClass ist eine Methode in der Klasse TimeTable. In dieser Methode werden die Stundenpläne der Klassen und der Schüler ausgegeben.

OutputTimetableofTeacher wie der Name schon aussagt wird hier der Stundenplan der Lehrer ausgegeben. Dies erfolgt mit den vom Benutzer ausgewählten Lehrer Variablen.

OutputTimetableofRoom wie in den zuvor beschriebenen Methoden passiert hier das selbe aber jetzt wird der Stundenplan des ausgewählten Raumes angezeigt

LoadData wird in der Klasse DataManager umgesetzt. Als erstes wird die Json Datei eingelesen und dann anschliessend in Statische Listen geschrieben. Sobald dies geschehen ist werden die ID's der Schüler und Lehrer wieder automatisch neu vergeben.

SafeData DataToSave als Datenobjekt wird im Ersten Schritt vorbereitet. In diesem Objekt sind die Listen aller Klassen, Lehrer und Räume. Anschliessend wird das Format der Json Datei schön lesbar gemacht. Sobald das getan worden ist, werden die Daten Serialisiert und dann Abgespeichert.

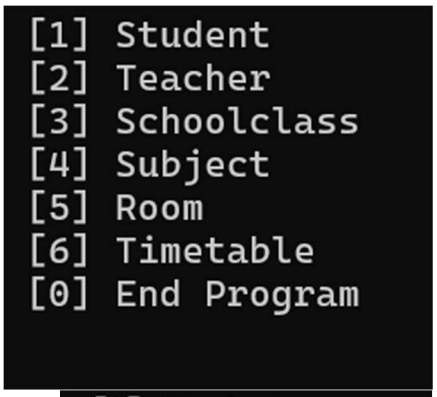
scoreClassPlan ist in der Klasse Evaluator. Im Ersten Schritt wird die Bewertung des Stundenplans einer Bestimmten Klasse auf den Normalwert 0 gesetzt. Danach werden die Freistunden bewertet. Als nächster Schritt werden die Raumwechsel bewertet. Als Vierter Schritt werden die Rand Stunden Morgens und Abends auch bewertet. Der Letzte Schritt ist das es einen Bonus gibt für Räume die nicht benutzt werden.

Build Zusammen Gefasst wird in dieser Methode einen neuen Stundeplan gemacht. Für die Verschiedenen Infos wie Lehrer und Raum wird einen Zusfallsgenerator verwendet. Für einen Simpleren Plan habe wir einfach die Klassen nach der Reihenfolge zugeordnet. Für jedes Fach sollte einen Slot gefunden werden und dann anschliessend wird dann noch jedes fach im Stundenplan Platziert. Dann wird geprüft ob für diese Stunde die Klasse bereites ein anderes Fach hat. Wenn dies gut ist, dann wird

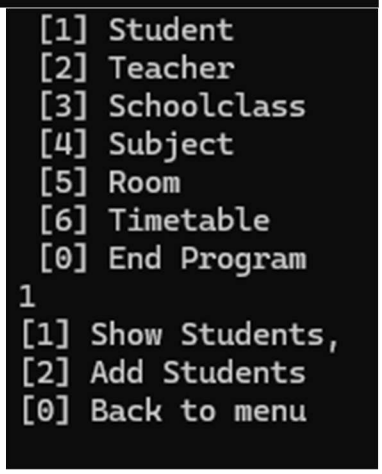
SwapSlots Hier werden die Klassen festgelegt bzw. ein Platzhalter. Es beginnt mit dem Slot an dem wir eine Temporäre Variable definieren bei uns ist dies d1 und h1. Im Nachhinein wird dann der Slot d1 und h1 mit d2 und h2 ersetz und zum Schluss kommt der Ursprüngliche Slot aus dem Ersten Schritt an die Stelle von d2 und h2. Aber wenn die Sachen besser sind als die alten wird der Slot fest gemacht.

Bedienungsanleitung

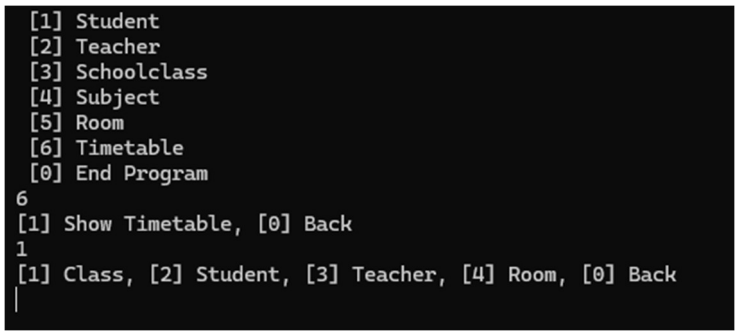
Man Beginnt im anfangs Menü und hat die Auswahl was man machen will in dem man die entsprechenden befehle schreibt.



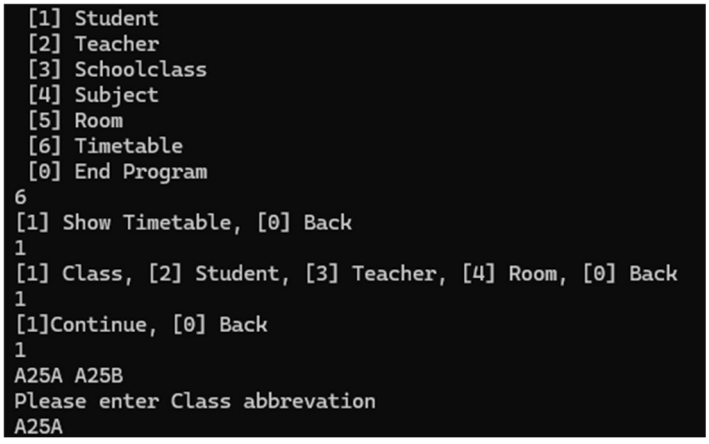
Bei den Meisten Auswahlen ist es etwa gleich aufgebaut und wird gleich bedient man hat beim Student, Teacher, Schoolclass, Room und Subject die Auswahl zwischen einem bestehend Objekt anzeigen und einem Neuen Objekt Hinzufügen.



Wenn man TimeTable ausgewählt hat, wird man gefragt von was man einen Stundenplan sehen möchte. Die Auswahl besteht zwischen Room, Teacher und Student.



Danach muss man sich für ein Spezifischen Stundeplan entscheiden, also von welchem Raum oder von welcher Person je nach dem was man vorher angewählt hat



Hier wird dann noch der Vorhin ausgewählte Stundenplan angezeigt. Dieser kann aber muss nicht voll sein.

Hier nur 2 Tage besetzt da nicht mehr Fächer ausgewählt worden sind.

Zeit	Mo	Di	Mi	Do	Fr
08:00	GP GOK A25A K103	SP KAM A25A K104			
09:00	DE KAM A25A K103	SP KAM A25A K103			
10:00		MA KAM A25A K101			
11:00	MA KAM A25A K102	BK GOK A25A K101			
12:00	EN KAM A25A K102	DE LAS A25A K104			
13:00					
15:00	GP GOK A25A K102	WR KAM A25A K104			
16:00	WR LAS A25A K102	WR KAM A25A K104			
17:00	EN KAM A25A K102	EN GOK A25A K104			

Qualitätskriterien & Bewertung

Wir haben mit Penaltys gearbeitet welche als Bewertung genutzt werden. Wir haben für alle Kriterien ein Penalty gemacht. Die Kriterien waren das man nicht so häufig das Zimmer wechseln soll, das nicht einfach komische Löcher im Stundenplan sind, Stunden die am Rand vom Tag sind werden schlechter bewertet ebenfalls haben wir auch Effiziente Raumnutzung belohnt. Es wird beim Stundenplan jeden einzelnen Tag bewertet und dann werden die schon erwähntet Kriterien geprüft und wenn was zutrifft zum Beispiel der Stunden Plan hat für jede Lektion ein anderes Zimmer wird es mit dem ScoreRoom den Penalty abgezogen.

In dem ScoreGaps werden die Lücken angeschaut als wenn die Schule erst in der 2 Lektion beginnt und davor keine Freistunde ist dann wird das Schlechter Bewertet.

```
private static int ScoreGaps(Schoolclass plan)
{
    int penalty = 0;

    for (int d = 0; d < Timetable.Days; d++)
    {
        TimetableSlot lastSeen = null;
        for (int h = 0; h < Timetable.Hours; h++)
        {
            TimetableSlot slot = plan.ClassPlan[d, h];

            if (slot != null)
            {
                if (lastSeen == null && h > 0)
                {
                    for (int x = 0; x < h; x++)
                    {
                        if (plan.ClassPlan[d, x] == null)
                        {
                            penalty -= Timetable.penalty_Gap;
                            break;
                        }
                    }
                }
                lastSeen = slot;
            }
        }
    }

    return penalty;
}
```

ScoreRoomChanges achtet auf die Nutzung der Räume und wenn es in der Vorherigen Stunde denn gleichen Raum verwendet wird dann wird es besser bewertet als wenn man den Raum wechseln muss.

```
private static int ScoreRoomChanges(Schoolclass plan)
{
    int penalty = 0;

    for (int d = 0; d < Timetable.Days; d++)
    {
        Room lastRoom = null;

        for (int h = 0; h < Timetable.Hours; h++)
        {
            TimetableSlot slot = plan.ClassPlan[d, h];
            if (slot == null) continue;

            if (lastRoom != null && slot.assignedRoom != lastRoom)
            {
                penalty -= Timetable.penalty_RoomChange;
            }

            lastRoom = slot.assignedRoom;
        }
    }

    return penalty;
}
```

Weil man nicht möchte das man am Beginn eine Stunde hat und danach frei für z.B 2 Lektionen oder am Schluss eine Stunde aber vorher Frei bis dort. Haben wir in dem ScoreEdgeHours es so definiert das es schlecht bewertet wird wenn nach der ersten Lektion Freistunden sind und vor der Letz

```
private static int ScoreEdgeHours(Schoolclass plan)
{
    int penalty = 0;

    for (int d = 0; d < Timetable.Days; d++)
    {
        if (plan.ClassPlan[d, 0] != null)
        {
            bool isolated = true;
            for (int h = 1; h <= 2 && h < Timetable.Hours; h++)
            {
                if (plan.ClassPlan[d, h] != null)
                {
                    isolated = false;
                    break;
                }
            }
            if (isolated) penalty -= Timetable.penalty_EdgeHour;
        }

        int last = Timetable.Hours - 1;
        if (plan.ClassPlan[d, last] != null)
        {
            bool isolated = true;
            for (int h = last - 1; h >= last - 2 && h >= 0; h--)
            {
                if (plan.ClassPlan[d, h] != null)
                {
                    isolated = false;
                    break;
                }
            }
            if (isolated) penalty -= Timetable.penalty_EdgeHour;
        }
    }
    return penalty;
```

ten Stunde Freistunden sind.

Als letztes Kriterium haben wir geschaut wenn die verschiedenen Klassen weniger oder gleich 90% die Räume besetzen, heisst so viel wie es sollte immer Räume geben die nicht besetzt sind weil man dann noch ein wenig Freiraum hat.

```
private static int ScoreResourceEfficiency()
{
    int score = 0;

    for (int d = 0; d < Timetable.Days; d++)
    {
        for (int h = 0; h < Timetable.Hours; h++)
        {
            List<Room> usedRooms = new List<Room>();

            foreach (Schoolclass classPlan in Schoolclass.AllClasses)
            {
                TimetableSlot slot = classPlan.ClassPlan[d, h];

                if (slot != null && slot.assignedRoom != null)
                {
                    usedRooms.Add(slot.assignedRoom);
                }
            }

            int numUsedRooms = usedRooms.Distinct().Count();

            if (numUsedRooms <= (int)Math.Round(Schoolclass.AllClasses.Count * 0.90, 0))
            {
                score += Timetable.penalty_RoomsUse;
            }
        }
    }
    return score;
```

Tests & Beispielsausgabe

Für einen Lehrer

Zeit	Mo	Di	Mi	Do	Fr
08:00	GP A25A GOK K103				
09:00	EN A25B GOK K104				
10:00					
11:00		BK A25A GOK K101			
12:00		WR A25B GOK K103			
13:00					
15:00	GP A25A GOK K102	EN A25B GOK K103			
16:00		SP A25B GOK K103			
17:00		EN A25A GOK K104			

Und hier für eine Klasse

A25A

Zeit	Mo	Di	Mi	Do	Fr
08:00	GP GOK A25A K103	SP KAM A25A K104			
09:00	DE KAM A25A K103	SP KAM A25A K103			
10:00		MA KAM A25A K101			
11:00	MA KAM A25A K102	BK GOK A25A K101			
12:00	EN KAM A25A K102	DE LAS A25A K104			
13:00					
15:00	GP GOK A25A K102	WR KAM A25A K104			
16:00	WR LAS A25A K102	WR KAM A25A K104			
17:00	EN KAM A25A K102	EN GOK A25A K104			

Reflexion & Teamarbeit

Jan:

Ich für meinen Teil der Arbeit bin mehr oder weniger damit zufrieden, was ich geleistet habe. Ich habe das Programm.cs sowie einige Darstellungen für die Dokumentation gemacht. Die Dokumentation war nicht so schwierig, jedoch das Programm war schon einiges Härter. Es war viel überlegen und Geduld gefordert. Das Ganze war aber alles im Gesamten ein interessantes Projekt, obwohl es viele Schwierigkeiten gab.

Es war auch grossartig nach einiger Zeit wieder ein C# Projekt machen zu können und dies auch umzusetzen. Dies ist nicht einfach aber immerhin möglich. Ich finde auch das solche Projekte für eine gute Erfahrung sorgen werden, denn in unserer Ausbildung mangelt es an Praktischer Übung. Daher bin ich froh dieses Projekt gemacht zu haben und die meine Zeit damit zu verbringen.

Juan:

Ich bin leider nicht so stark wie die anderen zwei wenn es ums Programmieren geht und deshalb habe ich mich um Sachen wie Diagramme Tabellen und Dokumentation gekümmert. Aber genau weil ich nicht so begabt bin wie die anderen Beiden habe ich sehr viel von diesem Projekt mitgenommen, da mir sehr viel erklärt und gezeigt wurde habe ich mehr Verständnis von dem Code und dem weg wie man das macht. Ich finde das unsere Teamarbeit sehr gut war und wir auch sehr gut uns verteilen konnten da wir unterschiedliche Stärken hatten.
Ich habe auch an dem Code gearbeitet aber hatte dort natürlich Schwierigkeiten aber ich habe das nach meiner Meinung Souverän gelöst beziehungsweise Wiktor hat mir das Souverän erklärt. Ich hatte aber sonst nicht so viel Schwierigkeiten und habe mich sehr wohl gefühlt mir der Arbeit, welche ich hatte. Wir haben unsere Zeit sehr gut eingesetzt und haben diese auch gut genutzt, deshalb bin ich froh, dass wir dieses Projekt zusammen bestritten haben und das ich so viel mitnehmen konnte.

Wiktor:

Auf das Projekt zurückblickend kann ich sagen, dass das Projekt viele Höhen und Tiefen hatte. Schon zu Beginn war mir bewusst, dass die Logik für die Erstellung des Stundenplans, das Herzstück des Projekts, am meisten Zeit in Anspruch nehmen würde. Dies ist auch der Grund, weshalb ich so früh wie möglich mit der Optimizer- und Evaluator-Klasse anfangen wollte, um bei Fehlern mehr Zeit zur Behebung zu haben.

Einer der Höhepunkte im Projekt war die erfolgreiche Implementierung des Algorithmus und die Erkenntnis, einen geeigneten Algorithmus gefunden zu haben, der nicht nur verständlicher ist als die anderen, sondern auch den Plan verbessert.

Ein großer Tiefpunkt war es am Ende, alle Teile zusammenzufügen, insbesondere die korrekte Integration der Datenpersistenz mit System.Text.Json, bei der sichergestellt werden musste, dass die statischen Listen und ID-Zähler nach dem Laden konsistent sind.

Ein unerwarteter, aber wiederkehrender Tiefpunkt im Projektmanagement war die Versionskontrolle mit Git. Es gab immer wieder Probleme beim Pushen von Änderungen und vor allem, wenn mehrere Leute am Projekt arbeiteten, wie es jeden Donnerstag der Fall war. Dies ist auch die Sache, die ich am meisten von diesem Projekt mitnehme, um bei zukünftigen Projekten diese Git-Probleme zu vermeiden.

Insgesamt hat das Projekt gezeigt, wie wichtig es ist, ein klares Klassenschema zu haben, um sich nicht im großen Wald von Klassen zu verlieren und so den Überblick über Zugriffe, Variablennamen und Speicherorte zu behalten. Und zudem hat das Projekt mir gezeigt, wie schwierig solch einfach klingende Projekte werden können.

Teamarbeit:
Wir haben von Beginn an gewusst das wir ein gutes Team bilden werden weil wir unterschiedliche Stärken und Schwächen hatten. Wir haben auch sehr gut unsere Stärken genutzt das wir unsere Schwäche gegenseitig Ausgleichen und das Beste aus uns raus holen konnten. Wir habe unsere Aufteilung so gemacht das Wiktor denn grössten Teil des Programmes zusammen mit Jan gemacht hat, Sie haben auch Juan gute Inputs und Erklärungen gegeben. Juan hat zum grössten Teil das Schriftliche übernommen aber auch die Diagramme und Darstellungen. Wir haben aus dieser Teamarbeit sehr viel mitgenommen besonders das man Leute mit verschiedenen Stärken braucht. Wir würden diese Gruppen-Konstellation für zukünftige Projekte wieder verwenden. Wir hatten was die Teamarbeit keine Probleme und haben uns relativ gut verstanden. Zusammengefasst finden wir haben sehr gut zusammen Agiert jeder hat seinen Part gemacht und wir sind sehr zufrieden.

Arbeitsrapport & Zeitaufwand

Datum	Wer	Aufgabe	Anzahl Lektionen	Bemerkung
23.10	Alle	Brainstorming	2	Wir hatten gute Ideen aber auch zu Komplexe
23.10	Jan	UML-Diagramm	1	
23.10	Juan	Sequenzdiagramm	1	
23.10	Wiktor	Github Aufsetzen	1	
30.10	Alle	Programmieren beginn	3	
06.11	Wiktor	TimeTable Programmiert	3	
06.11	Jan	Klassen implementiert	3	
06.11	Juan	Projektbeschreibung	2	
13.11	Wiktor	Logik beginn	3	Mergen von Jan und Wiktors Code machte Probleme
13.11	Juan	Use-Case Diagramm	3	
13.11	Jan	TimeTable Darstellung erstellt mit asci	2	
20.11	Juan	Bedienungsanleitung	3	
20.11	Jan	Programm.cs erstellen	4	Merge Probleme
20.11	Wiktor	Logik beendet	3	
27.11	Alle	Reflexion	1	
27.11	Alle	Feinschliff	1	
27.11	Alle	Abgabe		