

Energiewende Schweiz

Entwicklung einer Software-Applikation für die Dimensionierung von
Photovoltaik-Anlagen mit Batterie-Speicher

Semesterarbeit 2015/16
Software-Engineering & Projektmanagement
3. Semester – Klasse is3.3

«PV-Dimension»



Projektmitglieder:
Shane Hofstetter und Simon Müller

Inhaltsverzeichnis

1	ZUSAMMENFASSUNG	4
2	AUFGABENSTELLUNG (EINSCHUB)	5
3	PROJEKTMANAGEMENT	6
3.1	PROJEKTTEAM	6
3.2	ORGANIGRAMM	6
3.3	STRUKTURPLAN	7
3.4	ABLAUFPLAN	7
3.5	TERMINPLAN	8
4	SITUATIONSANALYSE	9
5	ZIELSETZUNGEN	11
5.1	SYSTEMZIELE	11
5.1.1	MUSS-ZIELE	11
5.1.2	WUNSCH-ZIELE	11
5.2	VORGEHENSZIELE	12
6	LÖSUNGSSUCHE	13
6.1	GROBKONZEPTENTWICKLUNG	13
6.1.1	TECHNOLOGIE	13
6.1.2	ANFORDERUNGEN	14
6.1.3	VERARBEITUNG UND BERECHNUNG	14
6.1.4	PHOTOVOLTAIK-ANLAGENTYP	14
6.1.5	STANDORT UND SONNENDATEN	14
6.1.6	GRANULIERUNG DER EINGABE- UND AUSGABEDATEN	15
6.2	KONZEPTANALYSE	15
7	LÖSUNGS-AUSWAHL	16
7.1	TECHNOLOGIE	16
7.2	STANDORT UND SONNENDATEN	17
7.3	BENUTZEREINGABEN UND UsABILITY	17
7.4	GRANULIERUNG DER EINGABE- UND AUSGABEDATEN	17
7.5	BERECHNUNGEN	17
7.6	PHOTOVOLTAIK-ANLAGENTYP	17
8	DETAILLKONZEPTENTWICKLUNG	18
8.1	ENTWICKLUNG KLASSENDIAGRAMM – GROBE SOFTWARE-ARCHITEKTUR	18
8.2	BERECHNUNGEN	19
8.2.1	BERECHNUNG DER PRODUZIERTEN LEISTUNG DES PV-GENERATORS	19
8.2.1.1	DIE WICHTIGSTEN GRÖSSEN ZUR KLASSIFIZIERUNG EINES PV-GENERATORS	19
8.2.1.2	BERECHNUNG DER DIREKTEN MODULEINSTRALUNG	19
8.2.1.3	SONNENWINKEL	20
8.2.1.4	IMPLEMENTIERUNG DER BERECHNUNGEN	22
8.3	BENUTZERSCHNITTSTELLE	22

8.3.1	ENTWURF DES DESIGNS	22
8.3.2	MVC-PATTERN	23
8.3.2.1	MODEL	23
8.3.2.2	VIEW	23
8.3.2.3	CONTROLLER.....	23
9	REALISIERUNG	24
9.1	BENUTZERSCHNITTSTELLE (GRAPHICAL USER INTERFACE)	24
9.1.1	PRINTSCREEN	24
9.1.2	AUFBAU	24
9.1.3	BENUTZER-EINGABEN.....	25
9.1.3.1	PV-GENERATOR	25
9.1.3.2	BATTERIE.....	25
9.1.3.3	SONNENDATEN.....	26
9.1.3.4	WIRTSCHAFTLICHKEIT	27
9.1.3.5	ERWEITERTE ANLAGEN-PARAMETER	28
9.1.3.6	KOMPONENTEN-DATENBANK.....	28
9.1.4	PROGRAMM-AUSGABEN	29
9.1.4.1	TAGESVERLAUF.....	29
9.1.4.2	WOCHENVERLAUF.....	29
9.1.4.3	WOCHENTOTAL	30
9.1.4.4	JAHRESÜBERSICHT.....	30
9.1.4.5	WIRTSCHAFTLICHKEIT	32
9.1.5	REALISIERUNG EINGABEFELDER	32
9.1.6	REALISIERUNG DIAGRAMME	33
9.2	PROGRAMM-LOGIK	34
9.2.1	BERECHNUNGEN	34
9.2.2	SIMULATION	34
9.2.2.1	SIMULATION MIT GEGEBENEN SONNENEINSTRALUNGSLEISTUNGEN.....	34
9.2.2.2	SIMULATION MIT SONNENEINSTRALUNGEN VOM WEBSERVICE	34
9.2.3	WEBSERVICES.....	36
9.2.3.1	KOORDINATEN.....	36
9.2.3.2	SONNENEINSTRALUNG	36
9.3	ZUSÄTZLICHE FEATURES	37
9.3.1	SPEICHERN UND LADEN.....	37
9.3.2	PDF-EXPORT	37
9.3.3	CSV-EXPORT.....	37
10	WIRTSCHAFTLICHKEIT	38
10.1	WIRTSCHAFTLICHKEIT DER APPLIKATION	38
10.2	WIRTSCHAFTLICHKEIT EINER PHOTOVOLTAIK-ANLAGE ANALYSIEREN	39
11	ANHANG	41
11.1	VERWENDETE HILFSMITTEL.....	41
11.2	QUELLENANGABEN	42
11.3	ANGEHÄNGTE DOKUMENTE.....	44

1 Zusammenfassung

Wir haben im Rahmen dieser Semesterarbeit den Auftrag erhalten, eine Software-Anwendung zur Dimensionierung und Ertragssimulation einer Photovoltaik-Anlage zu realisieren. Da die Photovoltaik-Technologie sich immer weiterverbreitet und mittlerweile auch für Privatpersonen erschwinglich ist, soll mit diesem Hilfsmittel der vermehrte Einsatz von PV-Anlagen unterstützt und der Planungsprozess vereinfacht werden.

Mit der Software sollen vor allem kleinere Photovoltaik-Anlagen, wie sie meist auf Dächern von Ein- oder Mehrfamilienhäusern realisiert werden, schnell und einfach simuliert werden können.

Die Berechnungsergebnisse der Software sollen aufzeigen, wie viel produzierter Strom für die gegebenen Anlagen-Parameter zu erwarten ist. Ausserdem soll anhand der Angabe eines Verbraucherprofils ermittelt werden können, ob und wieviel des produzierten Stromes direkt verbraucht werden kann.

Die entstandene Software haben wir mit der Programmiersprache Java sowie dem GUI¹-Framework JavaFX² umgesetzt. Dabei haben wir ein besonderes Augenmerk auf eine hohe Benutzerfreundlichkeit trotz grossem Funktionsumfang gelegt. Die Software soll für Privatpersonen ebenso wie für Fachplaner bedienbar sein.

Für die Sicherstellung der Praxistauglichkeit haben wir anhand von realen Verwendungsszenarien wichtige Funktionen ermittelt und implementiert, wie etwa den Export eines Projekt-Berichts nach abgeschlossener Anlagensimulation. Ausserdem lassen sich die eingetragenen Projekt-Daten als Datei speichern, sodass diese zu einem anderen Zeitpunkt wieder geladen werden können.

Die Angabe der Wetterdaten lässt sich ausserdem auf zwei Arten bewerkstelligen: Der Benutzer kann die Daten manuell eingeben, um die volle Kontrolle über die Berechnungen zu haben. Bei der zweiten Möglichkeit muss lediglich die Adresse des Anlagen-Standortes angegeben werden, die entsprechenden Sonnendaten werden dann von einer externen Datenbank abgerufen. Dies vereinfacht den gesamten Simulationsprozess noch weiter, da insgesamt weniger Benutzerdaten verlangt sind.

Aus der integrierten „Photovoltaik-Modul-Datenbank“ mit über 2000 Datensätzen lässt sich ausserdem direkt in der Anwendung ein geeignetes Modul suchen und für die Simulation übernehmen.

Um die Wirtschaftlichkeit der zu simulierenden Anlage zu prüfen, lassen sich ausserdem Strompreise im Ein- sowie Verkauf pro Kilowattstunde eintragen. Unsere Applikation errechnet den Jahresertrag der Anlage sowie die Menge Strom, welche nicht eingekauft werden musste. In einem weiteren Menü kann der Benutzer die Preise der Hardware-Komponenten eintragen, wie etwa für die Batterie und ein Solarpanel. Anhand dieser Daten wird die Amortisationsdauer errechnet und grafisch dargestellt.

¹ GUI = Graphical User Interface (Benutzerschnittstelle)

² JavaFX ist das moderne Framework zur Erstellung von grafischen Benutzeroberflächen

4 Situationsanalyse

Kaum ein Tag vergeht, ohne dass man in der Zeitung auf einen Artikel zum Thema Energiewende stösst. Auch wenn dieses Thema schon seit einigen Jahren aktuell ist, wird noch immer ca. 38% der Energie in der Schweiz mit Kernenergie erzeugt. Vom Bund werden laufend Massnahmen getroffen um von solchen fossilen Energien wegzukommen und vollständig auf erneuerbare Ressourcen zu setzen.

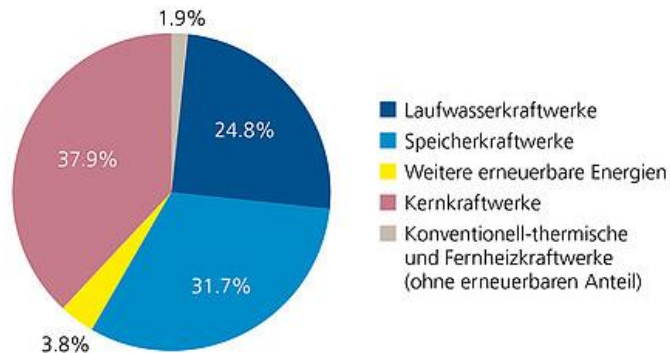


Abbildung 4.1 - Strommix Schweiz

Es gibt viele erneuerbare Energien, unter anderem: Windkraft, Kompogas, Wasserkraft und Photovoltaikanlagen (PVA). Die Wasserkraft (Laufwasser- sowie Speicherkraftwerke) stellen dabei knapp 60% der Stromproduktion. Die eher mager vertretene Photovoltaik-Energie befindet sich in den zusammengefassten 3.8% erneuerbarer Energien.

Wir sehen bei den aufgezählten erneuerbaren Energien vor allem bei den Photovoltaik-Anlagen noch viel Potenzial. Der Grund dafür ist relativ einfach: Photovoltaik-Anlagen können an sehr vielen Orten gebaut werden, eigentlich auf jedem Hausdach. Sie können beinahe beliebig dimensioniert werden und sind für Privatpersonen aus dem Mittelstand bezahlbar. Natürlich können Windkraft- und Kompogas-Anlagen auch einen grossen Beitrag zur Energiewende beisteuern, diese sind jedoch nicht für Privatpersonen erschwinglich.



Abbildung 4.2 - Montage einer Photovoltaik Aufdach-Anlage

Nach einigen Recherchen im Internet haben wir festgestellt, dass es kaum günstige Software für die Dimensionierung solcher Photovoltaik-Anlagen gibt. Weil solche Anlagen immer häufiger gebaut werden denken wir, dass sich eine Software gut verkaufen liesse welche die Dimensionierung und Ertragsberechnung solcher Anlagen erheblich vereinfacht.

Dies sehen wir als Chance eine gute, komfortable Software zu entwickeln. Mit dieser Software sollen auch Privatpersonen schnell und einfach Ertragssimulationen für eine Photovoltaik-Anlage auf ihrem eigenen Wohnhaus durchführen können. Die Applikation muss deshalb selbsterklärend sein. Für Unternehmen soll die Software die Planung einer solchen Anlage vereinfachen und somit Zeit und Kosten einsparen. Die Applikation soll deshalb auch professionellen Anwendern genug Funktionen bieten.

Damit unsere Software nach der ersten Implementierung immer auf dem aktuellen Stand bleiben kann, müssen wir bei der Entwicklung sehr darauf achten, dass wir einzelne Komponenten der Software unabhängig anpassen können. Ansonsten besteht die Gefahr, dass die Software mit der Zeit veraltet und sie beispielsweise wegen schlechter Nutzbarkeit auf Touchscreens niemand mehr benutzt.

Falls sich herausstellt, dass eine Mobile-Version der Applikation bei den Kunden sehr gut ankommen würde, sollten wir darauf gefasst sein. Deshalb ist es wichtig eine Technologie zu wählen, welche für solche Weiterentwicklungen portiert werden kann, um nicht den gesamten Entwicklungsaufwand erneut zu haben. Ein weiteres Mittel den Zeit- und Kostenaufwand einzugrenzen und gleichzeitig ein breites Einsatzspektrum der Applikation zu gewährleisten ist der Einsatz einer plattformunabhängigen Technologie.

Ein Risiko bei der Entwicklung dieser Applikation stellt weiterhin das Zeit- und Ressourcenmanagement dar. Da es sich um eine Software handelt, bei welcher wir uns neue Kompetenzen aneignen müssen, lässt sich der gesamte Zeit- und somit Kostenaufwand nur grob abschätzen.

5 Zielsetzungen

5.1 Systemziele

Es soll ein Hilfsmittel für Privatpersonen ebenso wie für Photovoltaik-Planer erstellt werden.

Mit diesem Hilfsmittel soll es kostengünstig und ohne vertieftes Verständnis der zugrunde liegenden Solartechnologie möglich sein, eine übersichtliche Ertragssimulation für eine Photovoltaik-Anlage anhand von eingegebenen Parametern durchzuführen.

Die Lösung, bzw. Art dieses Hilfsmittels ist im Rahmen der Semesterarbeit vorgegeben und sieht eine Software-Applikation vor.

5.1.1 Muss-Ziele

Mit der Software-Applikation soll auf einfache, jedoch realistische Weise eine Ertragssimulation erstellt werden. Die Simulation erfolgt anhand der vom Benutzer eingegebenen Daten. Damit soll der Entscheidungsprozess erleichtert werden, ob sich eine Photovoltaik-Anlage auf dem vorgesehenen Dach bzw. der vorgesehenen Fläche energietechnisch lohnt. Das Programm soll selbständig anhand der eingegebenen Parametern die nötigen Berechnungen durchführen, um die gewünschten Ausgabewerte für den Zeitraum von einer Woche zu bestimmen. Die Resultate sollen in einem Diagramm übersichtlich dargestellt werden.

5.1.2 Wunsch-Ziele

1. Ein weiter entferntes Ziel könnte sein, die Anlagen in der Software auf wirtschaftliche Rentabilität zu prüfen.
2. Die Sonneneinstrahlung für den Standort der PV-Anlage soll anhand der Adresse ermittelt werden.
3. Für Photovoltaik-Anlagen mit Nutzung des produzierten Stroms (Eigennutzung) soll ausserdem die nötige Kapazität der eingesetzten Speicherbatterie errechnet werden.
4. Die eingegebenen Daten sollen feiner granuliert werden können, beispielsweise für jeden Tag stündlich. Dies würde eine genauere und realistischere Ausgabe erzeugen.
5. Die eingegebenen Daten sollen als Projekt-Datei gespeichert und bei Bedarf wieder mit der Software geöffnet werden können.
6. Die Applikation kann einen Projektbericht erzeugen. Dieser könnte zum Beispiel als PDF gespeichert werden können.
7. Es kann eine Hilfeseite angezeigt werden, wo die einzelnen Eingabe-Parameter und ihren Einfluss auf die Berechnungen erklärt werden.
8. Die Berechnung wird automatisch auf das ganze Jahr ausgedehnt. Somit kann der Planungsprozess einer PV-Anlage noch vereinfacht werden, weil weniger Simulationen erstellt werden müssen.

5.2 Vorgehensziele

Die Kosten für das Projekt sollten angesichts der Tatsache, dass wir die Applikation einer breiten Masse günstig zur Verfügung stellen möchten, möglichst tief gehalten werden. Die Entwicklungskosten sollten daher den Betrag von CHF 15'000 nicht überschreiten.

Der Fertigstellungstermin ist durch den Terminplan im Rahmen der Semesterarbeit bereits gegeben.

6 Lösungssuche

6.1 Grobkonzeptentwicklung

6.1.1 Technologie

Die zu erstellende Software-Anwendung kann mit den unterschiedlichsten Technologien erstellt werden, wobei jede Technologie ihre Vor- sowie Nachteile hat. Obwohl die zu verwendende Technologie im Rahmen des Fachs Software-Engineering bereits gegeben ist, wollen wir diesen sehr wichtigen Schritt nicht vergessen und zählen hier kurz die Möglichkeiten auf:

1. Desktop-Anwendung

a. Java mit JavaFX oder Swing

Plattformunabhängige Anwendung, objekt-orientierte Programmierung.



Abbildung 6.1 -JavaFX Logo

b. C# mit WPF³

Windows-Anwendung, objekt-orientierte Programmierung.



Abbildung 6.2 - WPF Logo

c. Weitere

2. Web-Anwendung

Die Applikation könnte auch als Web-Anwendung erstellt werden, was ebenfalls eine Plattformunabhängigkeit bieten würde. Ausserdem gibt es keinen Installationsaufwand.

3. Mobile-App

Die Anwendung könnte ebenfalls als Mobile-App für Android und/oder iOS erstellt werden. Mit solch einer mobilen Anwendung könnte der Planer direkt beim Kunden eine kurze Berechnung durchführen.

Die gewählten Plattformen schliessen sich nicht zwingend aus, es kann beispielsweise eine Desktop- sowie eine Mobile-Anwendung erstellt werden. Ebenfalls liesse sich eine Kombination aus Web-Anwendung und Mobile-App nicht ausschliessen.

³ WPF = Windows Präsentation Format, das moderne Framework von Microsoft zur Erstellung von Benutzerschnittstellen für die Windows-Plattform.

6.1.2 Anforderungen

Die genauen Anforderungen mit den Ein- sowie Ausgaben des Programms sind bereits in der Aufgabenstellung aufgelistet. Wir wollen die Eingaben nochmals übersichtlich darstellen mit genauer Bedeutung sowie physikalischer Einheit.

Folgende Eingaben sind aus der Aufgabenstellung entnommen:

EINGABE	BEDEUTUNG / EINHEIT
ANZAHL DER PANEL	Anzahl Stück der Photovoltaik-Module
FLÄCHE DER PANEL	Fläche eines eingesetzten Moduls [m ²]
AUFSTELLUNG DER PANEL	Aufstellung der Panel [° Grad]
AUSRICHTUNG DER PANEL	Ausrichtung (Azimut) der Panel [° Grad]
WIRKUNGSGRAD	Modulwirkungsgrad eines Panels [0..1]
KAPAZITÄT DER SPEICHERBATTERIE	Speicherkapazität [kWh]
PROFIL DER SONNENEINSTRALUNG	Einstrahlungsstärke pro Fläche [W/m ²]
PROFIL DER ENERGIE NUTZUNG	Benötigte Leistung [W]

Weitere Eingaben, welche wir für eine Berechnung des Ertrags als zwingend erachten:

EINGABE	BEDEUTUNG / EINHEIT
MONAT DER BERECHNUNG	Für welchen Monat im Jahr der Ertrag berechnet wird. Für die Berechnung des Ertrags hat die Sonnenbahn einen grossen Einfluss.

6.1.3 Verarbeitung und Berechnung

Die Berechnung der produzierten Leistung der Photovoltaik-Anlage hat nach anerkannten Methoden zu erfolgen. Das Ergebnis der Berechnungen, bzw. ihre Richtigkeit ist in erster Linie an die Genauigkeit der vom Benutzer gegebenen Daten gebunden.

6.1.4 Photovoltaik-Anlagentyp

Es gibt verschiedene Anlagentypen: Die netzgekoppelten Anlagen verbrauchen einen Teil des produzierten Stromes selber (Eigenverbrauch). Bei netzautarken Anlagen (vorzufinden in entlegenen Regionen) muss der Strombezug vollständig aus der Batterie erfolgen, der Strom kann nicht in das Netz eingespeist werden. Wir können uns entweder entscheiden, die Berechnungen für einen der beiden Anlagentypen durchzuführen oder beide Varianten zu unterstützen.

6.1.5 Standort und Sonnendaten

Nach eingehender Recherche bezüglich der Photovoltaik-Technologie hat sich ergeben, dass der Standort der PV-Anlage einen sehr entscheidenden Faktor im Zusammenhang mit der produzierten Leistung darstellt. Deshalb scheint einer der wichtigsten Eingaben des Benutzers der Verlauf der tatsächlichen Sonneneinstrahlung zu sein.

Zusätzlich kommt erschwerend dazu, dass diese Daten ohne Zugriff auf spezielle Wetter-Datenbanken wie beispielsweise «Meteonorm» schwer herauszufinden sind.

Da es sich um standortabhängige Daten handelt, könnten diese auch durch die Eingabe der Adresse des Anlagen-Standortes von einer öffentlichen Datenbank abgefragt werden. Dies würde die Simulation für den Benutzer relativ stark vereinfachen, sowie gleichzeitig die Qualität der Resultate erhöhen. Diese Methode setzt jedoch einen passenden Webservice mit einer vertrauenswürdigen Datenbank voraus.

Es ergeben sich dadurch zwei Konzepte für die Beschaffung des Profils zur Sonneneinstrahlung:

1. Manuelle Eingabe des Profils durch den Benutzer.
2. Eingabe der Standort-Adresse und darauffolgende Beschaffung der Sonnendaten von einem Webservice.

6.1.6 Granulierung der Eingabe- und Ausgabedaten

Einen Einfluss auf die Verwendbarkeit sowie Realitätsnähe der errechneten Daten stellt ebenfalls die Genauigkeit der Datenreihen dar. In der Aufgabenstellung ist von einem Wert pro Tag die Rede. Der Benutzer könnte die Daten aber auch mit kleineren Zeit-Deltas zur Verfügung stellen und würde daher im Gegenzug eine feinere Ausgabe erhalten.

Es ergeben sich hier mehrere Möglichkeiten zur Ein- sowie Ausgabe der Daten, wir zählen hier kurz drei realistische auf:

1. Tages-Basis (Ein Wert pro Tag)
2. Stunden-Basis (Ein Wert pro Stunde, eines Tages)
3. Viertelstunden-Basis (Ein Wert pro Viertelstunde, eines Tages)

6.2 Konzeptanalyse

Da das grobe Konzept der Lösung (eine Desktop-Software zu schreiben mit Java) bereits gegeben ist, haben wir Lösungsvarianten für detaillierte Ausprägungen gesucht.

Zum Kapitel 6.1.5 – Standort und Sonnendaten:

Die Lösungsvariante, welche die Eingabe des Anlagen-Standorts vorsieht, um damit die Wetterdaten von einem Webservice abzufragen, befriedigt nicht zwingenderweise das Muss-Ziel, die Wetterdaten eingeben zu können. Sie lassen sich hierbei indirekt eingeben, der User hat allerdings keine volle Kontrolle. Wir sollten hier deshalb vorsichtig sein. Es wäre am besten, wenn beide Varianten unterstützt werden.

Alle anderen Lösungsvarianten würden die Muss-Ziele vollumfänglich erfüllen.

7 Lösungsauswahl

7.1 Technologie

Die Vorgabe der Schule ist es, eine Java Anwendung zu entwickeln. Somit sind die Möglichkeiten der Technologie schon stark eingegrenzt. Trotzdem gibt es noch einige Entscheide zu fällen. Zum einen, ob wir für das Graphical User Interface (GUI) die schon ältere Swing- oder die neuere JavaFX-Bibliothek einsetzen sollen und zum anderen, welche externen Java Bibliotheken wir verwenden sollen.

Der grosse Vorteil von Java ist, dass die Applikation damit plattformunabhängig wird. Die Software liesse sich dann auf jedem Computer ausführen, auf welchem das Java Runtime Environment (JRE) installiert ist.

JavaFX bringt vor allem auch, weil wir für die Datenausgaben verschiedene Charts benötigen werden, gegenüber «awt⁴» und «Swing» grosse Vorteile mit sich:

- Integrierte Charts Klassen (keine externe Bibliothek und kein grosser Programmieraufwand notwendig)
- Läuft auch auf Mobilgeräten (theoretisch)
- Schöneres und moderneres „Look and Feel“
- Wird immer noch weiterentwickelt
- Modernere API

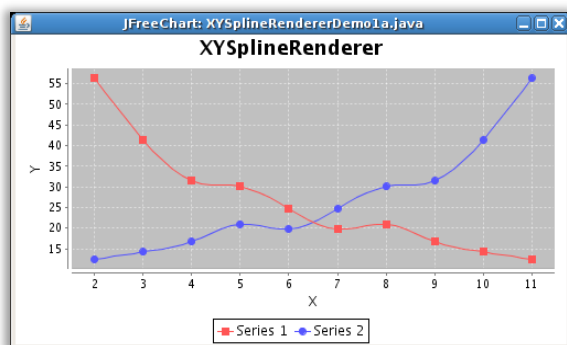


Abbildung 7.1 - Eine mit der JFreeChart Bibliothek generierte Line-Chart

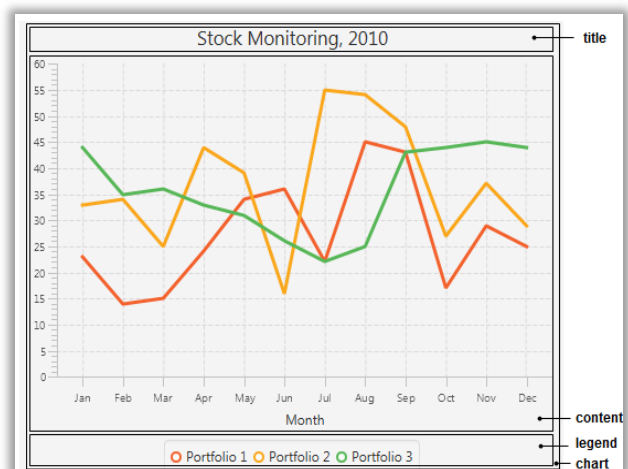


Abbildung 7.2 - Eine mit der JavaFX generierte Line-Chart

Vor allem wegen den integrierten Chart-Klassen, aber auch wegen den anderen Punkten, haben wir uns entschieden das GUI mit JavaFX zu implementieren.

Da eine Applikation ohne Datenausgabe im Druckformat wenig Sinn ergibt, implementieren wir die Möglichkeit die Ergebnisse in Form eines Projekt-Berichts als PDF zu exportieren. Weil es in den Java eigenen Bibliotheken keine einfache Möglichkeit dazu gibt, haben wir uns dazu entschieden die Java Bibliothek «itextpdf» zu verwenden.

⁴ Abstract Window Toolkit, Java API zur Erstellung von plattformunabhängigen Benutzeroberflächen mittels nativen GUI-Komponenten

Damit die eingegebenen Daten gespeichert und bei Bedarf wieder geladen werden können, werden wir mit der Serialisierungs-Funktion von Java die Möglichkeit implementieren die Anlagen-Parameter als Datei abzuspeichern.

7.2 Standort und Sonnendaten

Wir werden primär die Möglichkeit für den Benutzer implementieren, die Sonnendaten (oder Wetterdaten) manuell einzugeben. Des Weiteren werden wir, falls die Zeit reicht, ebenfalls die Möglichkeit der automatischen Ermittlung der Wetterdaten anhand des Anlagenstandorts implementieren.

7.3 Benutzereingaben und Usability

Für die Benutzereingaben verwenden wir verschiedene Elemente der «JavaFX»-Bibliothek, damit trotz einer relativ grossen Datenmenge die eingegeben werden muss das GUI übersichtlich bleibt, verwenden wir für die Dateneingabe genauso wie für die Datenausgabe Diagramme.

Wir werden auch die Möglichkeit implementieren, die Adresse vom PV-Anlagen-Standort eingeben zu können. Die Sonneneinstrahlungsdaten werden dann über einen Webservice geholt. Dazu wird allerdings eine Internetverbindung benötigt.

7.4 Granulierung der Eingabe- und Ausgabedaten

Damit die Berechnungsergebnisse die Realität möglichst gut abbilden, haben wir uns dafür entschieden, diese stündlich zu ermitteln, somit kann der User den Stromverbrauch für jede Stunde und jeden Tag in der Woche eingeben. Weil dies aber die einzugebende Datenmenge erhöht, haben wir entschieden dem User auch die Möglichkeit zu geben nur den Stromverlauf eines Tages einzugeben, und diesen dann für die ganze Woche zu verwenden.

7.5 Berechnungen

Die Formeln zur Berechnung des Energie-Ertrages aus den gegebenen Anlage-Daten werden wir aus der entsprechenden Fachliteratur entnehmen. Die Berechnungen werden in der Detailkonzeptentwicklung weiter erläutert.

7.6 Photovoltaik-Anlagentyp

Aus den zwei möglichen Anlagentypen (netzautark und netzgebunden) werden wir die Software primär auf netzgebundene Anlagen ausrichten. Dies deshalb, weil diese Anlagen sehr viel häufiger gebaut werden. Die netzautarken Anlagen finden sich vor allem in abgelegenen Regionen ohne Stromnetz. Der Unterschied in der Software ist nicht gross, die Texte bei den Programm-Ein- sowie Ausgaben werden ein wenig anders sein.

In einem optionalen Schritt könnte auch beim Start des Programms oder in den Optionen eine Auswahl getroffen werden, um welchen Anlagentyp es sich handelt.

8 Detailkonzeptentwicklung

8.1 Entwicklung Klassendiagramm – Grobe Software-Architektur

In einem ersten Schritt werden wir ein UML (Unified Modelling Language) Diagramm erstellen um die grobe Struktur der Applikation festzulegen. Weil es sich um eine Hardware-Installation handelt (Solar-Panel, Wechselrichter, Batterie, etc.) lassen sich die jeweiligen Klassen relativ gut bestimmen und ableiten.

Es wird also für die verschiedenen Hardware-Komponenten eine Klasse geben, in welchen dieselben Komponenten konkretisiert werden. Eine Photovoltaik-Anlage besteht aus mehreren Solar-Panels, welche in einem Feld angeordnet sind. Wir erstellen deshalb eine Klasse für das Solar-Panel-Feld. Des Weiteren halten wir es für sinnvoll, eine Klasse für die komplette Anlage (den PV-Generator) zu haben.

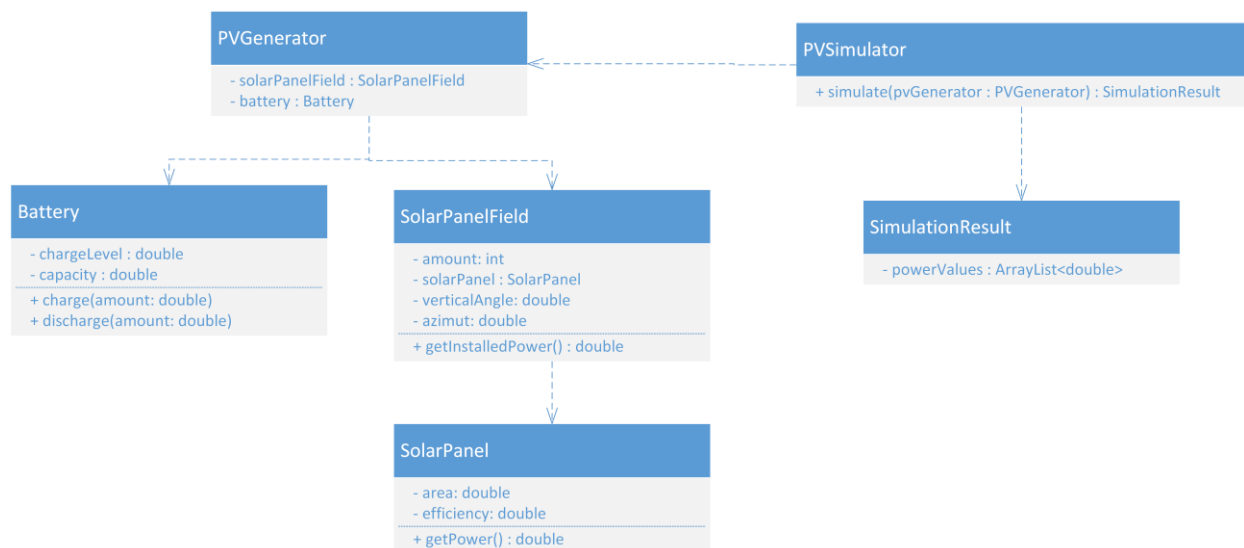


Abbildung 8.1 - Entwurf UML-Diagramm für den Aufbau der Software

Mit der PV-Generator Klasse lässt sich die komplette Photovoltaik-Anlage in einem Objekt unterbringen, welches Solar-Panel-Feld, Batterie und Wechselrichter als Member-Variablen enthält. Eine Simulator-Klasse soll dann die nötigen Berechnungen vornehmen, um den Ertrag der Anlage sowie den Stromverbrauch zu bestimmen. Damit dies gelingt, müssen die benötigten Daten (Sonneneinstrahlung, Stromverbrauch) vorhanden sein. Wir werden diese in einer eigenen Klasse unterbringen, damit ausser dem Leistungswert noch weitere Daten wie Zeit-Stempel abgelegt werden können.

8.2 Berechnungen

In diesem Kapitel geht es darum, aufzuzeigen wie wir den Ertrag der Photovoltaik-Anlage anhand der gegebenen Benutzerdaten berechnen können.

8.2.1 Berechnung der produzierten Leistung des PV-Generators

8.2.1.1 Die wichtigsten Grössen zur Klassifizierung eines PV-Generators

Total Anlagenleistung:

$$P_{tot} [W_P] = n_{panel} * P_{panel}$$

Total Modulfläche:

$$A_{PV} [m^2] = n_{panel} * A_{panel}$$

Installierbare Leistung:

$$P_{inst} [W] = \eta_{panel} * A_{PV} * 1000 \frac{W}{m^2}$$

8.2.1.2 Berechnung der direkten Moduleinstrahlung

Die Sonneneinstrahlungswerte auf Strahlungskarten sind im Normalfall als Einstrahlung auf die horizontale Ebene angegeben [Wh/m² oder W/m²]. Dieser Wert entspricht nicht der Einstrahlung welche die Sonne bei 90° Einstrahlungswinkel auf die Horizontale erreicht.

Da die produzierte Leistung nicht direkt mit diesem Wert berechnet werden kann, da er nicht die Einstrahlung auf das Panel beschreibt, muss der Einstrahlungswert umgerechnet werden.

Die Situation ist auf folgender Grafik ersichtlich:

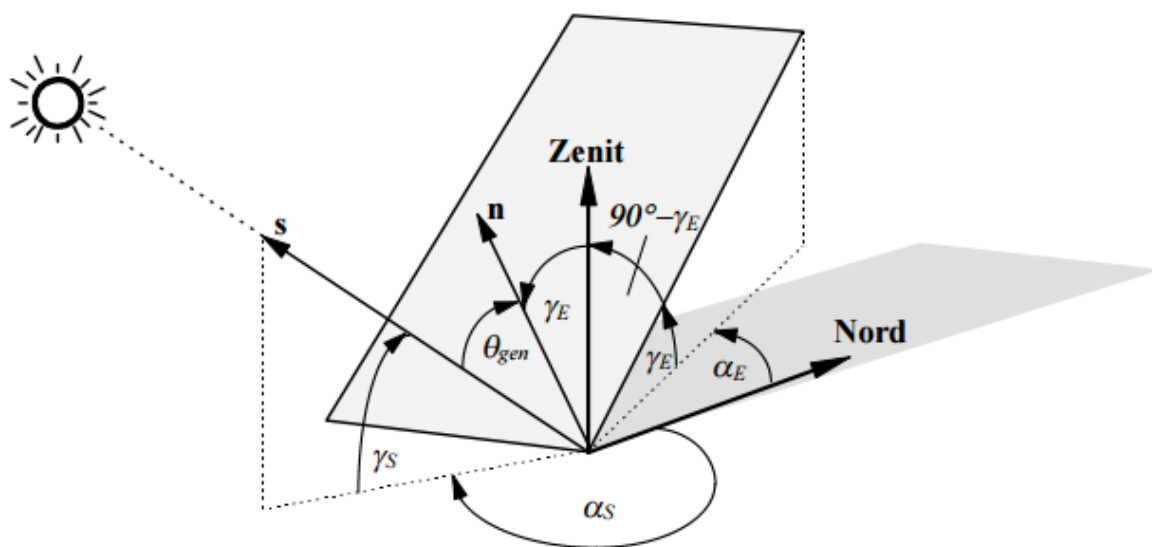


Abbildung 8.2 - Zusammensetzung des Modulwinkels

Je nach Aufstellungs- sowie Ausrichtungswinkel kann die Einstrahlung verbessert oder verschlechtert werden gegenüber dem angegebenen horizontalen Messwert. Der zunächst gesuchte Wert ist der Generator-Einfallswinkel θ_{gen} , dieser beschreibt den Winkel zwischen der Sonne und der Modulfläche. Er wird wie folgt berechnet:

$$\begin{aligned}\theta_{gen} &= \arccos(\mathbf{s} \cdot \mathbf{n}) = \\ &= \arccos(-\cos \alpha_S \cdot \cos \gamma_S \cdot \cos \alpha_E \cdot \sin \gamma_E - \\ &\quad \sin \alpha_S \cdot \cos \gamma_S \cdot \sin \alpha_E \cdot \sin \gamma_E + \sin \gamma_S \cdot \cos \gamma_E) = \\ &= \arccos(-\cos \gamma_S \cdot \sin \gamma_E \cdot \cos(\alpha_S - \alpha_E) + \sin \gamma_S \cdot \cos \gamma_E)\end{aligned}$$

Abbildung 8.3 - Berechnung des Generator-Winkels

Benötigt werden also folgende Winkel:

- Modul-Aufstellwinkel γ_E
- Modul-Azimut α_E
- Sonnen-Azimut α_S
- Sonnenhöhenwinkel γ_S

Die Modulwinkel werden dabei vom Benutzer in die Applikation eingegeben.

8.2.1.3 Sonnenwinkel

Für die Berechnung des Generator-Einfallswinkels werden Daten zum Verlauf der Sonnenbahnen benötigt. Diese sind stark standortabhängig und sehen für Berlin folgendermassen aus:

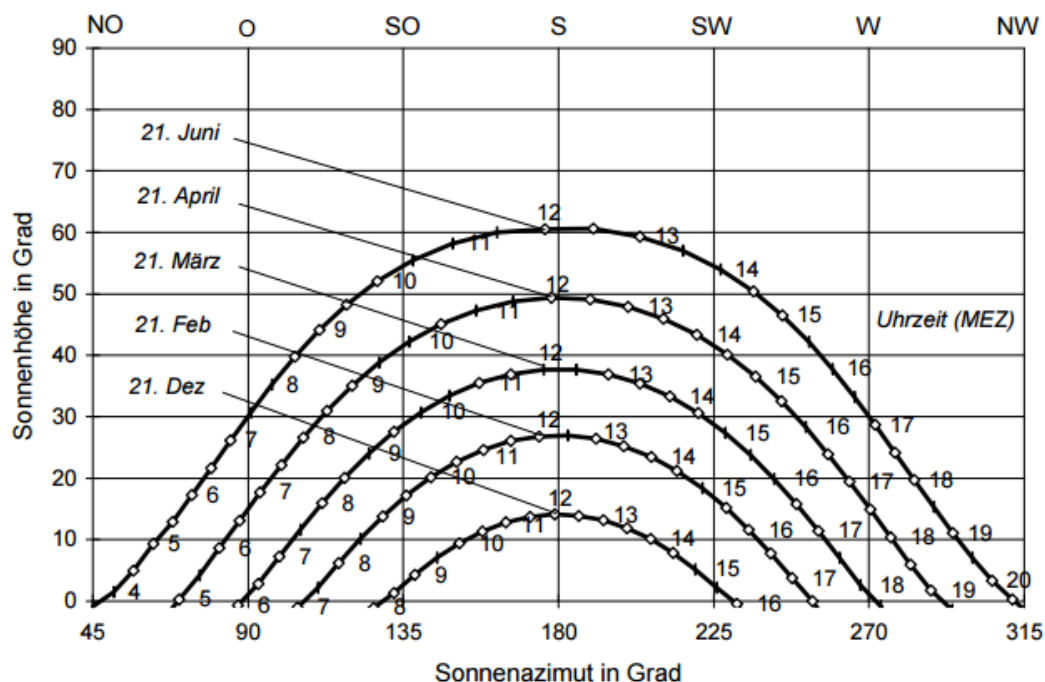


Abbildung 8.4 - Sonnenwinkel-Jahreszeit

Für den Offline-Betrieb unserer Applikation haben wir den Verlauf der Sonnenbahn für jeden Monat im Jahr für den Standort Baden lokal gespeichert. Wir rechnen mit eher moderaten Abweichungen innerhalb der Schweiz.

Die Daten sind in einfachen Text-Dateien für jeden Monat gespeichert und sehen folgendermassen aus:

April
Uhrzeit Azimut Altitude
06:30 72.17 0
06:45 74.94 1.14
07:00 77.67 3.61
07:15 80.40 6.10
...

Mit den vorhandenen Daten lässt sich nun für eine bestimmte Tageszeit der Generator-Einfallswinkel θ_{gen} berechnen. Dieser ist wie bereits erwähnt jedoch nur ein Zwischenziel, wir wollen nun die tatsächlich eintreffende Strahlung auf dem Modul $E_{dir,gen}$ anhand der horizontalen Einstrahlung $E_{dir,hor}$ berechnen:

$$E_{dir,gen} = E_{dir,hor} \cdot \left\{ \max\left(0, \frac{\cos\theta_{gen}}{\sin\gamma_s}\right) \right\}.$$

Abbildung 8.5 Direkte Generator-Einstrahlungsenergie

Die horizontale Einstrahlung $E_{dir,hor}$ wird mit einem Faktor, der sich aus dem Generator-Einfallswinkel θ_{gen} sowie der Sonnenhöhe γ_s ergibt, multipliziert.

Mit der direkten Einstrahlung $E_{dir,gen}$ auf den PV-Generator lässt sich nun die produzierte Leistung E_{real} sowie E_{ideal} errechnen.

Idealer Ertrag:

$$E_{ideal} \left[\frac{kWh}{a} \right] = \eta_{panel} * A_{PV} * E_{dir,gen}$$

Realer Ertrag:

$$E_{real} \left[\frac{kWh}{a} \right] = PR * n_{verschattung} * E_{ideal}$$

$$PR = Performance Ratio$$

Die Performance Ratio sowie der Verschattungsgrad sind angenommene Werte. Die Performance Ratio lässt sich nur sehr schwer theoretisch berechnen, weil es der Wirkungsgrad des kompletten Systems samt Modulen, Wechselrichter, etc. ist. Sie wird erst während dem Betrieb der Anlage aus den Messwerten errechnet.

Der Verschattungsgrad beschreibt den Verlust an Sonneneinstrahlung durch schattenerzeugende Objekte wie Bäume in der Umgebung des PV-Generators.

Der Benutzer soll in einem Menü diese beiden Koeffizienten manuell bestimmen können.

8.2.1.4 Implementierung der Berechnungen

Die Berechnungen werden wir in speziell dafür vorgesehenen, unabhängigen Klassen implementieren. Ein Teil der Formeln wird dabei in Form von statischen Funktionen realisiert sein.

Die PV-Anlage, beziehungsweise dessen abstrahiertes Objekt soll in einer Ertragssimulator-Klasse alle Berechnungsschritte durchlaufen, bis die Resultate erzeugt sind. Die Berechnungs-Resultate sollen dabei wiederum in Form eines komplexen Objekts zurückgegeben werden. Dies wird die Ausbaufähigkeit der Software gewährleisten.

8.3 Benutzerschnittstelle

8.3.1 Entwurf des Designs

Das Ziel ist eine Applikation mit einer guten Usability zu programmieren. Daher ist es sehr wichtig, dass wir die vielen Benutzereingaben welche auf dem GUI gemacht werden können gut strukturieren. Wir möchten der Übersichtlichkeit halber nur ein einziges Hauptfenster anzeigen, auf welchem die komplette Funktionalität verfügbar ist. Auf diesem Fenster sehen wir eine räumliche Trennung der Benutzerein- sowie -Ausgaben vor. Dazu haben wir einen Entwurf erstellt welchen wir während der Entwicklung als grobe Vorlage benutzen können.

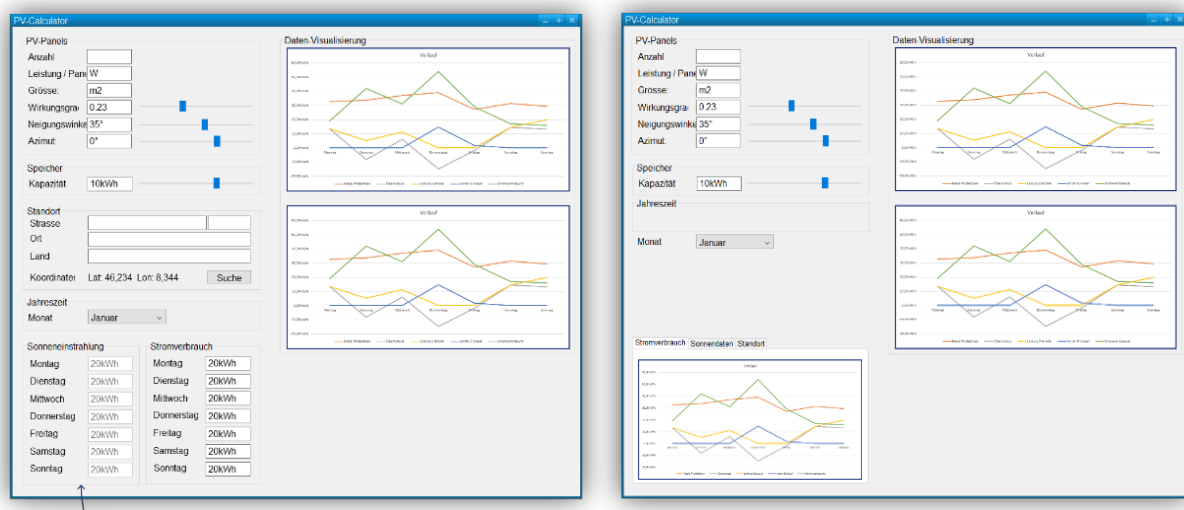


Abbildung 8.6 – Entwurf des GUI-Designs

Wir haben zunächst ein Design entwickelt, bei welchem die Benutzereingaben mittels Eingabefeldern und «Slidern» vorgenommen werden. Damit aber nicht zu viele Zahlen über die Tastatur eingetippt werden müssen, haben wir beschlossen, dass die Eingaben so weit wie sinnvoll und möglich mit der Maus gemacht werden sollen. Deshalb haben wir ein zweites Design erstellt, bei welchem interaktive Linien-Diagramme verwendet werden. Der Benutzer könnte hier die Punkte im Diagramm mit der Maus ändern und somit bequem Leistungsprofile eingeben. Wir werden versuchen, die Variante mit den interaktiven Diagrammen umzusetzen. Falls es jedoch Probleme bei der Implementierung gibt, werden wir auf die erste Variante ausweichen.

Damit der Benutzer die Wetterdaten des Anlage-Standorts nicht selber recherchieren und eintragen muss, möchten wir die Möglichkeit zur Ermittlung der Daten über einen Webservice implementieren. Der Benutzer kann somit die Adresse eintragen und die Daten werden direkt von einer Datenbank geladen, sofern der Rechner über eine funktionierende Internet-Verbindung verfügt. Dies werden wir jedoch wie im Kapitel Lösungsauswahl bereits dokumentiert, nur bei ausreichenden zeitlichen Ressourcen implementieren.

8.3.2 MVC-Pattern

Wir werden in der Applikation so weit als möglich versuchen, das MVC-Pattern (Model-View-Controller) zu verwenden. Was wir dabei primär umsetzen möchten, ist die strikte Trennung von Datenein/-Ausgabe, Verarbeitung und Speicherung.

8.3.2.1 *Model*

Die Speicherung der Daten erfolgt in den Models, diese werden unsere PV-Anlage-typischen Klassen sein (Batterie, Solarpanel, usw.).

8.3.2.2 *View*

Mit der View ist die eigentliche Benutzer-Ansicht gemeint. Wir erstellen für fast jedes Datenmodel eine eigene View-Klasse. Diese Klassen werden in ihrem Konstruktor⁵ die GUI-Ansicht erzeugen.

8.3.2.3 *Controller*

Die Controller-Klasse verbindet ein Model mit der zugehörigen View. Die Daten werden vom Model geholt und ausgabebereit an die View weitergegeben. In die andere Richtung werden Elemente in der View mit «Event-Listnern» ausgestattet und es wird auf die Ereignisse reagiert. Gegebenenfalls werden Benutzerdaten in den Models gespeichert.

⁵ Prozedur zum Erstellen eines Objekts.

9 Realisierung

9.1 Benutzerschnittstelle (Graphical User Interface)

9.1.1 Printscreen

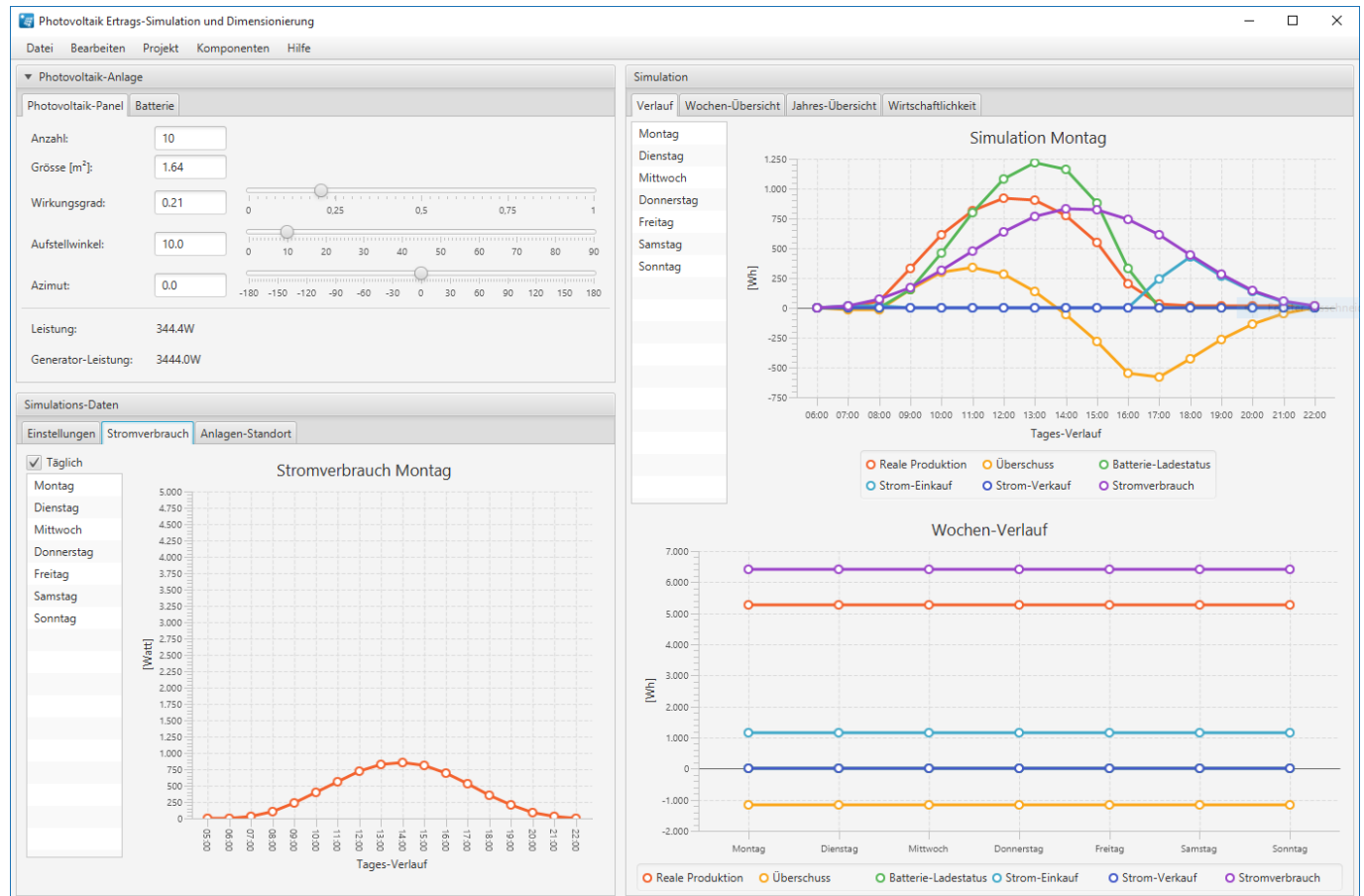


Abbildung 9.1 Screenshot der Applikation

9.1.2 Aufbau

Wir haben uns entschieden, die Kontrollelemente nach Eingabe und Ausgabe zu unterteilen. Die Eingabe-Elemente wie PV-Generator Angaben und Leistungskurven befinden sich auf der linken Seite des Bildes. Alle Simulationsergebnisse bzw. dessen Diagramme befinden sich rechts im Bild.

Da die Kontrollelemente insgesamt relativ viel Platz benötigen, wir aber dennoch alles in einem Hauptfenster anzeigen wollen, haben wir uns für eine Organisation mittels Tabs entschieden. Es wäre auch eine Lösung mittels Scrollbalken oder mehreren Fenstern denkbar. Wir sind aber mit unserer Lösung mit der jetzigen Funktionsausstattung sehr zufrieden. Falls die Software eines Tages noch weiter ausgebaut werden sollte, muss man die Organisation eventuell nochmals überdenken. Da die Kontrollelemente jedoch gut abgekapselt sind, lässt sich eine Layout-Änderung mit kleinem Aufwand bewerkstelligen.

9.1.3 Benutzer-Eingaben

In diesem Kapitel wird erläutert, welche Parameter verändert werden können und wo diese Einstellungen vorgenommen werden.

9.1.3.1 PV-Generator

Die wichtigsten Angaben zur simulierten Anlage finden sich im Reiter „Photovoltaik-Panel“. Hier werden alle nötigen Parameter eingegeben um die installierte Anlage zu spezifizieren.

Parameter	Value
Anzahl	10
Grösse [m²]	1.64
Wirkungsgrad	0.21
Aufstellwinkel	10.0
Azimut	0.0
Leistung	344.4W
Generator-Leistung	3444.0W

Abbildung 9.2 - Einstellungen zum Photovoltaik-Panel

Die Leistung eines Panels sowie die Gesamtleistung der Anlage werden dabei sofort errechnet und an entsprechender Stelle ausgegeben. Die Leistung eines Panels ergibt sich dabei vom Wirkungsgrad sowie der Fläche eines Panels. Die Gesamtleistung der Anlage ist schliesslich die Summe aller Panel-Leistungen.

9.1.3.2 Batterie

Im Reiter „Batterie“ kann die Speicher-Kapazität der Batterie festgelegt werden. Die Auswirkungen sind in den Simulations-Ergebnissen sofort ersichtlich.

Parameter	Value
Kapazität [kWh]	5.0
Anfangs-Ladezustand [kWh]	0.0

Abbildung 9.3 - Einstellungen zur Batterie

Ausserdem lässt sich der Anfangs-Ladezustand der Batterie bestimmen, dieser beschreibt die gespeicherte Energie am Anfang der Simulation.

9.1.3.3 Sonnendaten

Die Eingabe der Leistungskurve der Sonneneinstrahlung lässt sich auf zwei Arten realisieren:

- Manuelle Eingabe
- Automatische Ermittlung

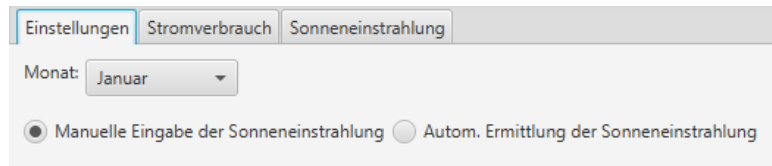


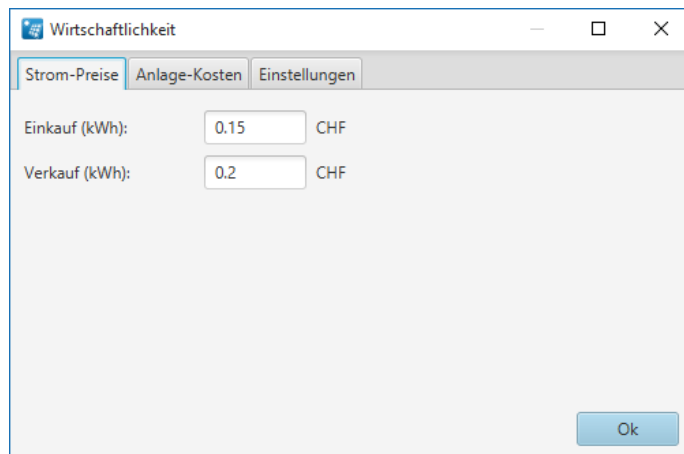
Abbildung 9.4 - Simulationseinstellungen

Die gewünschte Methode lässt sich über den entsprechenden Radio-Button auswählen. Dabei wird die Ansicht verändert, sodass die entsprechenden Kontrollelemente sichtbar sind.

Ausserdem lässt sich der Monat bestimmen, für welchen die Wochensimulation gelten soll. Bei der Entwicklung der Anwendung war diese Angabe zu Beginn sehr wichtig, da aufgrund dieses Monats die entsprechenden Daten geladen wurden. Ab Implementierung der Jahres-Simulation wird aber nun das komplette Jahr durchgerechnet, es werden aber in der Wochen-Ansicht die Ergebnisse des angegebenen Monats dargestellt.

9.1.3.4 Wirtschaftlichkeit

Unter dem Menüpunkt Projekt / Wirtschaftlichkeit lassen sich diverse ökonomische Parameter einstellen. Im Reiter „Strom-Preise“ lassen sich Strom-Einkaufs sowie Verkaufspreis eintragen. Des Weiteren lässt sich im Reiter „Anlage-Kosten“ der Preis für jede Komponente der Photovoltaik-Anlage einstellen. In den Einstellungen ist es ausserdem noch möglich, die angezeigte Währung anzupassen. Einstellungen werden bei Klick auf „Ok“ übernommen.



The screenshot shows a window titled 'Wirtschaftlichkeit' with three tabs: 'Strom-Preise', 'Anlage-Kosten', and 'Einstellungen'. The 'Strom-Preise' tab is active. It contains two input fields: 'Einkauf (kWh):' with a value of 0.15 and 'Verkauf (kWh):' with a value of 0.2. Both fields have 'CHF' as the currency. An 'Ok' button is located at the bottom right.

Parameter	Value	Unit
Einkauf (kWh)	0.15	CHF
Verkauf (kWh)	0.2	CHF

Abbildung 9.5 - Einstellungen Strom-Preise



The screenshot shows the same 'Wirtschaftlichkeit' window, but with the 'Anlage-Kosten' tab active. It contains four input fields: 'Solar-Panel:' with a value of 250.0, 'Batterie:' with a value of 2000.0, 'Wechselrichter:' with a value of 2000.0, and 'Planung und Installation:' with a value of 4000.0. All fields have 'CHF' as the currency. An 'Ok' button is located at the bottom right.

Parameter	Value	Unit
Solar-Panel	250.0	CHF
Batterie	2000.0	CHF
Wechselrichter	2000.0	CHF
Planung und Installation	4000.0	CHF

Abbildung 9.6 - Einstellungen Anlage-Kosten

9.1.3.5 Erweiterte Anlagen-Parameter

Unter diesem Menüpunkt lassen sich spezielle Anlagen-Parameter einstellen, sie richten sich an erfahrene Benutzer und Photovoltaik-Experten. Die einstellbaren Faktoren Performance-Ratio sowie Verschattungsfaktor greifen direkt in die Ertrags-Berechnungen ein und sollten somit vorsichtig verändert werden.

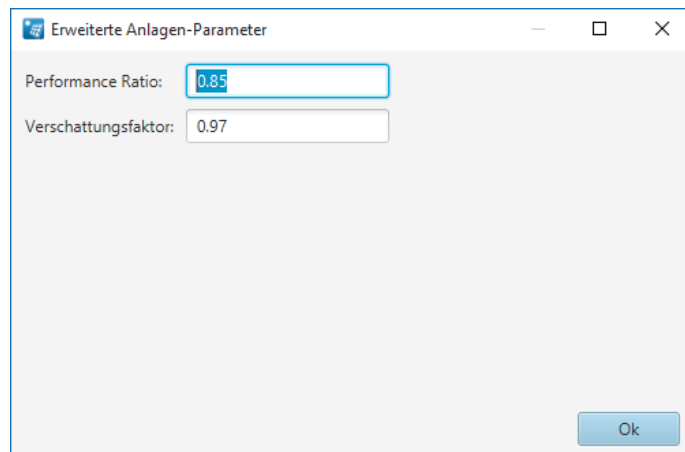
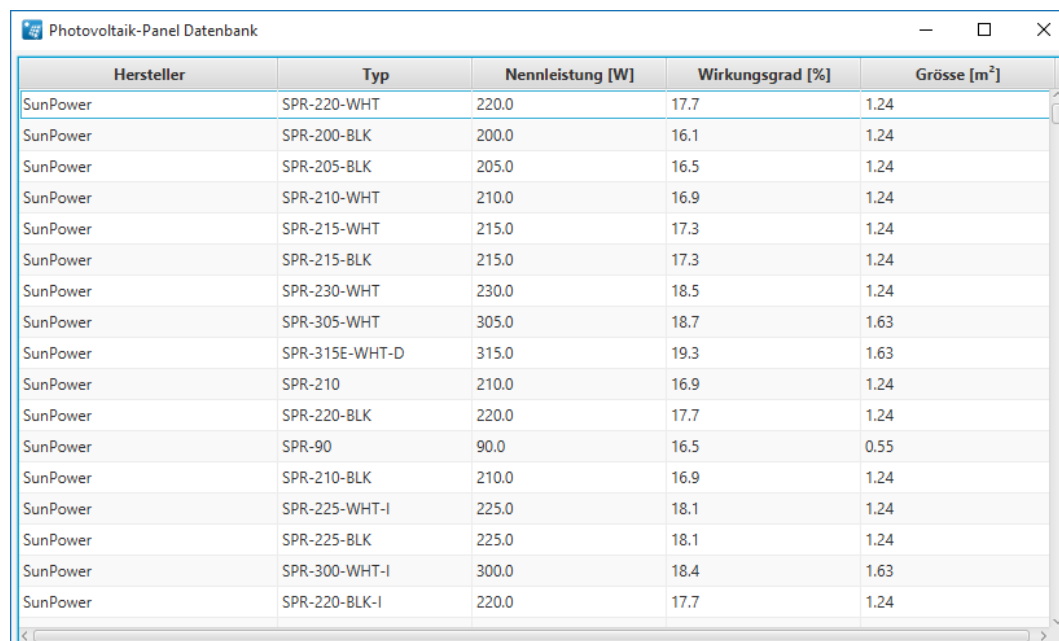


Abbildung 9.7 - Erweiterte Anlagen-Parameter

9.1.3.6 Komponenten-Datenbank

Wir haben zu den Produkten der bekanntesten Photovoltaik-Hersteller die Spezifikationen zusammengetragen und daraus eine Solarpanel-Datenbank erstellt. Sie befindet sich unter dem Menüpunkt Komponenten / Solar-Panels. Die Tabelle umfasst ungefähr 2000 verschiedene Photovoltaik-Panels. Mit einem Rechtsklick auf ein Panel und dem anschliessenden Betätigen des entsprechenden Menüpunktes lassen sich die Daten in die Applikation übernehmen und simulieren.



Hersteller	Typ	Nennleistung [W]	Wirkungsgrad [%]	Grösse [m²]
SunPower	SPR-220-WHT	220.0	17.7	1.24
SunPower	SPR-200-BLK	200.0	16.1	1.24
SunPower	SPR-205-BLK	205.0	16.5	1.24
SunPower	SPR-210-WHT	210.0	16.9	1.24
SunPower	SPR-215-WHT	215.0	17.3	1.24
SunPower	SPR-215-BLK	215.0	17.3	1.24
SunPower	SPR-230-WHT	230.0	18.5	1.24
SunPower	SPR-305-WHT	305.0	18.7	1.63
SunPower	SPR-315E-WHT-D	315.0	19.3	1.63
SunPower	SPR-210	210.0	16.9	1.24
SunPower	SPR-220-BLK	220.0	17.7	1.24
SunPower	SPR-90	90.0	16.5	0.55
SunPower	SPR-210-BLK	210.0	16.9	1.24
SunPower	SPR-225-WHT-I	225.0	18.1	1.24
SunPower	SPR-225-BLK	225.0	18.1	1.24
SunPower	SPR-300-WHT-I	300.0	18.4	1.63
SunPower	SPR-220-BLK-I	220.0	17.7	1.24

Abbildung 9.8 - Solarpanel-Datenbank

9.1.4 Programm-Ausgaben

Nach dem Durchlaufen der Simulationsberechnungen werden diverse Ertragsdaten im Zeit-Verlauf dargestellt. Es kann der Tagesverlauf für jeden Tag einer Woche im eingestellten Monat angesehen werden sowie das daraus resultierende Wochen-Total. Des Weiteren wird aufgrund der lokal vorhandenen Sonnenbahn-Daten oder ggf. der Daten vom Webservice die Simulationsergebnisse des gesamten Jahres dargestellt. Aufgrund des Jahresertrags findet anschliessend eine Berechnung der Amortisationszeit statt, dessen Ergebnisse werden ebenfalls jahresweise dargestellt. Mit dieser Fülle an Daten kann die simulierte Anlage umfangreich evaluiert werden.

9.1.4.1 Tagesverlauf

Im Reiter „Verlauf“ in den Simulationsergebnissen lässt sich der Tagesverlauf anzeigen, welcher anhand der eingestellten Parameter entsteht. Die Wichtigsten Grössen sind die reale Produktion sowie der Stromverbrauch. Aus diesen beiden Werten werden die restlichen Grössen gebildet. In der Liste auf der linken Seite kann der Tagesverlauf des entsprechenden Tages angezeigt werden.

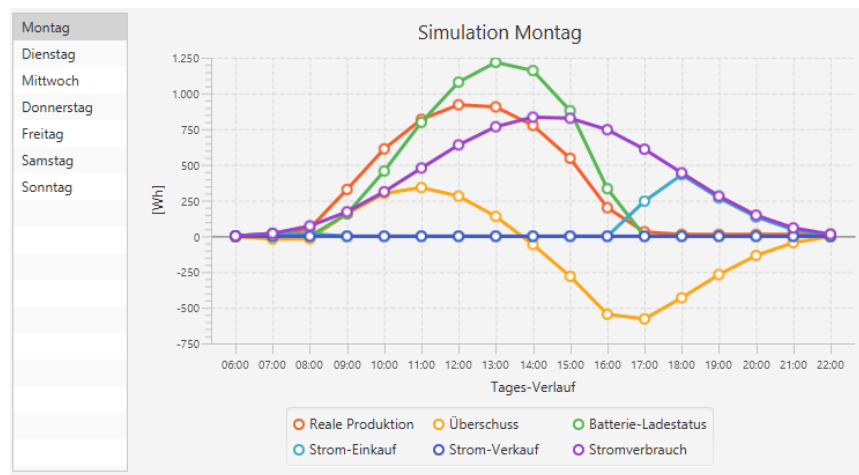


Abbildung 9.9 - Tagesverlauf der Simulations-Ergebnisse

9.1.4.2 Wochenverlauf

Aus den kumulierten (aufsummierten) Werten der Tages-Verläufe wird anschliessend der Wochen-Verlauf gebildet und angezeigt. Er beinhaltet dieselben Grössen wie der Tages-Verlauf.

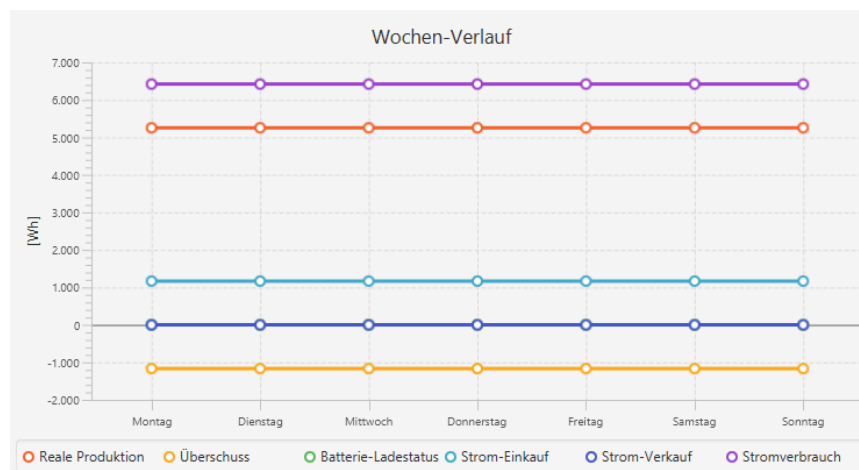


Abbildung 9.10 - Wochenverlauf der Simulations-Ergebnisse

9.1.4.3 Wochentotal

In der „Wochen-Übersicht“ werden zusätzlich alle Ergebnisse aus dem Wochen-Verlauf aufsummiert und als Balken-Diagramm dargestellt. Hier sind ausser der physikalischen Werte ebenfalls ökonomische Beträge ersichtlich, welche sich aus der Produktion sowie den eingestellten Strom-Ein- sowie Verkaufspreisen ergeben. Bei diesem und den weiteren Balkendiagrammen lässt sich der numerische Wert anzeigen, indem der Mauszeiger über den entsprechenden Balken geführt wird.

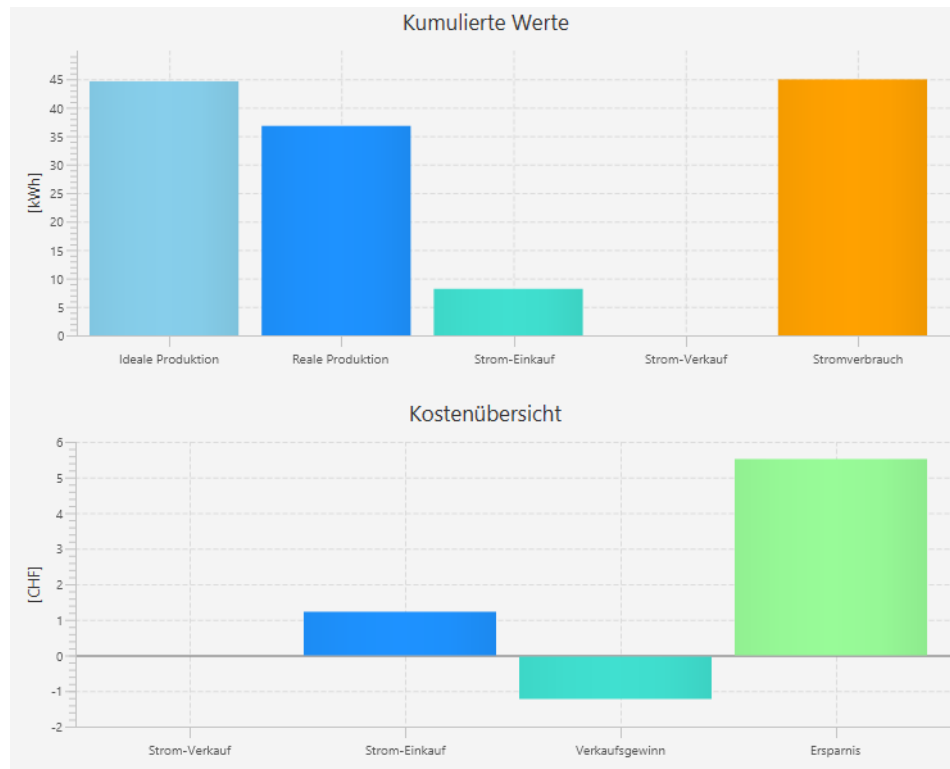


Abbildung 9.11 - Wochen-Total

9.1.4.4 Jahresübersicht

In der „Jahres-Übersicht“ sind die Anlagen-Werte für jeden Monat aufsummiert und in der Grössenordnung eines Jahres dargestellt. Dies halten wir für die interessanteste Ausgabe, hier lassen sich die Auswirkungen der verschiedenen Anlagen-Parameter am Besten analysieren und beurteilen.

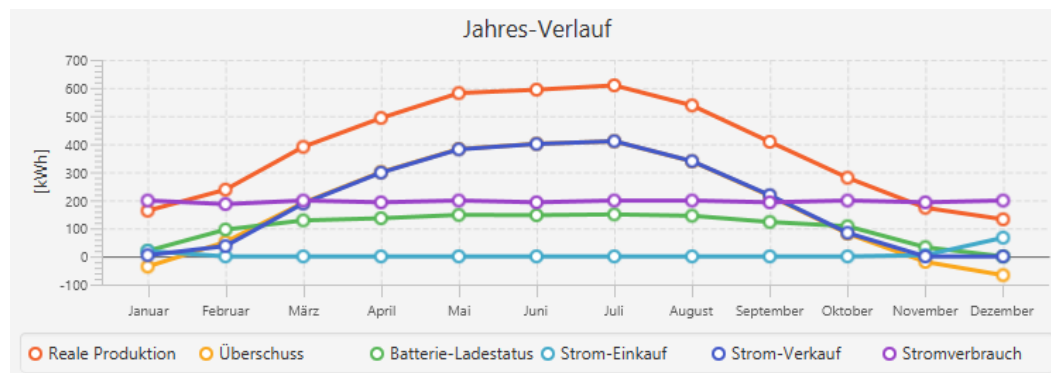


Abbildung 9.12 - Jahresverlauf

Die Werte jedes Monats werden anschliessend ebenfalls aufsummiert und als Jahres-Total in einem Balkendiagramm dargestellt. Hier kann die gesamte Produktion eines Jahres abgelesen werden.

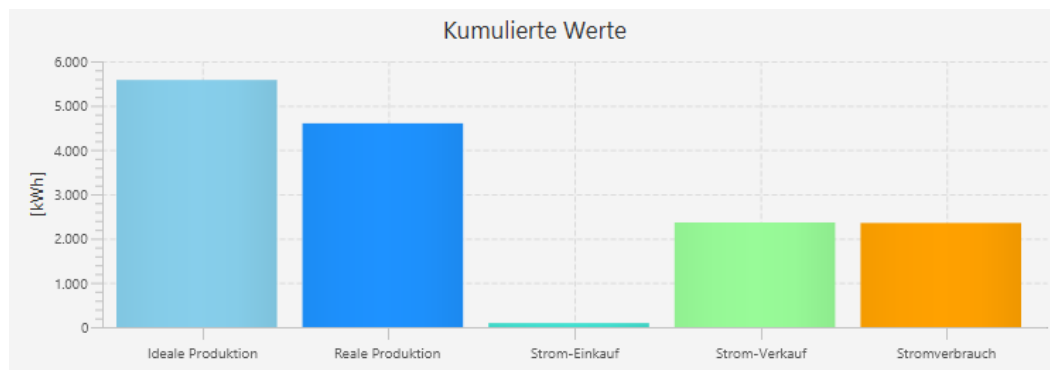


Abbildung 9.13 - Kumulierte Ergebnisse des gesamten Jahres

Zu guter Letzt werden die ökonomischen Werte aufsummiert und ebenfalls in einem Balken-Diagramm dargestellt. Hier lässt sich aus dem Verkaufsgewinn des Stromes sowie der Kosten-Ersparnis (Strom der wegen des Eigenverbrauchs nicht eingekauft werden musste) der Ertrag der Anlage über den Zeitraum eines Jahres berechnen.

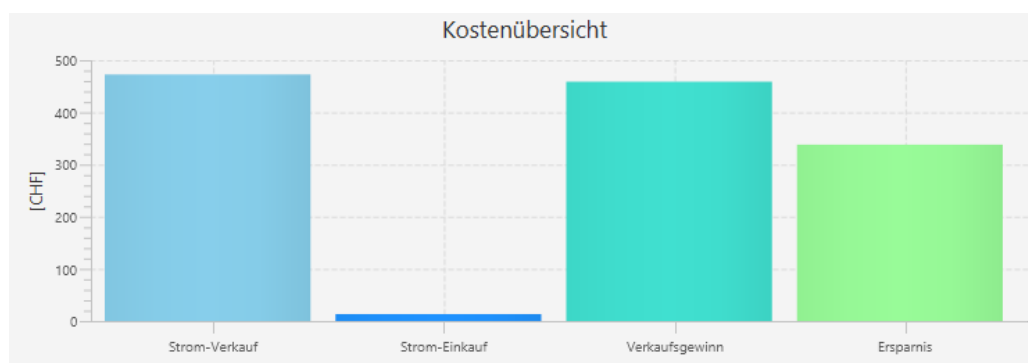


Abbildung 9.14 - Kumulierte Gewinne und Kosten des gesamten Jahres

9.1.4.5 Wirtschaftlichkeit

Im Reiter „Wirtschaftlichkeit“ werden die Ergebnisse der Amortisationsrechnung präsentiert. Beim Ertrag handelt es sich um den Gewinn bzw. die Ersparnis, welche/r Total von der Anlage generiert wurde. Von den totalen Anlage-Kosten wird der Ertrag abgezogen und das Ergebnis im Diagramm als „Restkosten“ dargestellt. Es lässt sich somit den ungefähren Zeitpunkt ablesen, ab wann die Anlage ihre Kosten gedeckt hat.

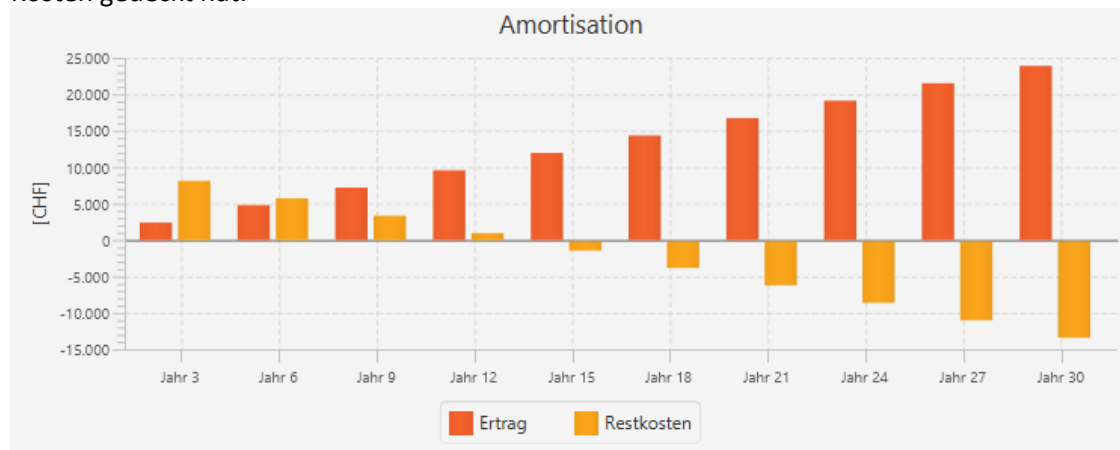


Abbildung 9.15 - Amortisationszeit

9.1.5 Realisierung Eingabefelder

Wir haben für die numerischen Angaben eigene Eingabefeld-Klassen erstellt, beziehungsweise die Standard-Klassen erweitert. Vor allem ist die «IntegerTextField» sowie die «DoubleTextField»-Klasse sehr hilfreich, um Benutzereingaben zu validieren. Diese lassen nur den jeweiligen Datentyp als Eingabe zu.

Anzahl:	<input type="text" value="10"/>	
Grösse [m²]:	<input type="text" value="1.64"/>	
Wirkungsgrad:	<input type="text" value="0.21"/>	<input type="range" value="0.21"/>
Aufstellwinkel:	<input type="text" value="10.0"/>	<input type="range" value="10.0"/>
Azimet:	<input type="text" value="0.0"/>	<input type="range" value="0.0"/>

Abbildung 9.16 - Ausschnitt eines Screenshots der Applikation

Des Weiteren haben wir für eine bequeme Änderung der Simulations-Bedingungen die Klasse «InputWidget» erstellt, welche ein Text-Feld sowie einen «Slider» als Schnittstelle zur Verfügung stellt. Somit lässt sich entweder eine genaue Dateneingabe über das Textfeld realisieren, oder aber eine schnelle, ungefähre Eingabe über den «Slider». Die beiden Werte sind aneinander gekoppelt und weisen einen Minimal- sowie Maximalwert auf.

9.1.6 Realisierung Diagramme

Für die Eingabe der Leistungsdaten durch den Benutzer direkt in ein Linien-Diagramm, haben wir die «LineChart»-Klasse von JavaFX erweitert, sodass diese interaktiv ist. Es handelt sich im Projekt um die «ExtendedLineChart»-Klasse, welche von der «LineChart»-Klasse erbt.

Diese Klasse, welche noch nicht projektspezifisch ist, haben wir für die konkrete und bequeme Verwendung im Projekt nochmals erweitert, um ein höheres Abstraktionslevel zu bieten. Viele Konfigurationen, wie etwa die Interaktivität, werden in diesen konkreten Klassen bereits vorgenommen. Dies lässt sich so implementieren, weil diese Diagramme nun bereits einen definierten Verwendungszweck haben und sehr spezifisch sind. Wir haben bei der Erweiterung der «ExtendedLineChart» -Klasse nach Eingabe- sowie Ausgabe-Diagrammen unterschieden.

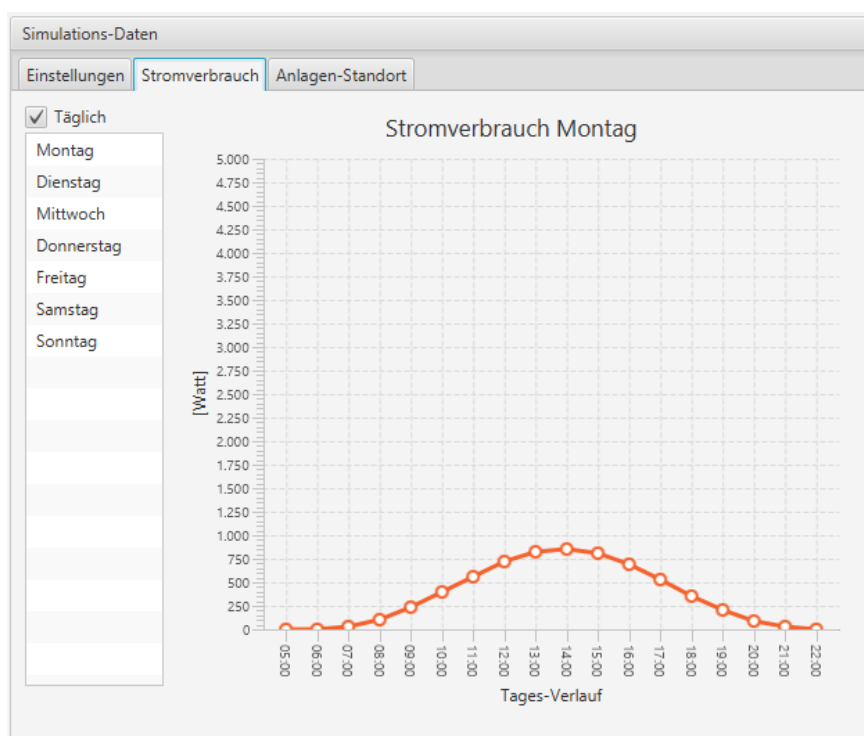


Abbildung 9.17 - Interaktives Liniendiagramm zum Eingeben des Stromverbrauchs

Diese spezifischen Charts verwenden wir beispielsweise in der Wochenansicht, wobei zum Diagramm noch eine Liste mit den Wochentagen hinzukommt. Mit der Auswahl eines Wochentages wird dem Diagramm das entsprechende Daten-Paket übergeben. Durch die hohe Abstraktion lässt sich dies relativ einfach bewerkstelligen.

9.2 Programm-Logik

9.2.1 Berechnungen

Für einfachere Berechnungen mit primitiven Datentypen haben wir die Klasse „PVMath“ erstellt, welche diverse statische Funktionen zur Verfügung stellt. Die Simulation der PV-Anlage wird in der „PVWeekSimulator“-Klasse ausgeführt. Aus dieser Klasse muss ein Objekt erstellt werden, welchem alle zur Simulation benötigten Daten-Objekte übergeben werden müssen.

9.2.2 Simulation

Der Simulator führt eine komplette Ertragssimulation einer Woche auf Stunden- oder Tages-Basis durch. Je nach eingestelltem Sonnendaten-Modus (manuell oder automatisch) holt sich der Simulator selbstständig die benötigten Daten vom entsprechenden Service.

Die Ertragsberechnung mit manuell eingegebenen Daten ist hierbei wegen zusätzlichen Berechnungen aufwändiger als die automatische Methode. Wie die Sonnendaten geholt werden, ist in den nächsten beiden Unterkapiteln beschrieben.

Mit den ermittelten Generator-Leistungen wird dann für jeden Zeitpunkt die produzierte Leistung errechnet. Danach kann anhand vom momentanen Verbrauch berechnet werden, wieviel Strom in die Batterie geladen wird bzw. wieviel sie entladen wird. Ist die Batterie vollständig geladen wird der Überschuss in das Stromnetz eingespeist. Falls die Batterie jedoch vollständig entladen ist und es wird weiter Strom benötigt, wird dieser vom Netz bezogen bzw. eingekauft.

9.2.2.1 *Simulation mit gegebenen Sonneneinstrahlungsleistungen*

Die Leistung der Sonnenstrahlung wird vom Benutzer für jede Zeiteinheit gegeben. Da es sich beim gegebenen Wert um die horizontale Einstrahlung handelt, muss diese in die direkte Generator-Einstrahlung umgerechnet werden. Die Formeln dazu sind im Kapitel Berechnungen in der Detailkonzeptentwicklung zu finden.

Für die Berechnung der Generator-Einstrahlung werden die Sonnenwinkel für diese Zeit benötigt. Wir haben für den Standort Baden innerhalb des Software-Projekts lokale Dateien gespeichert, in welchen die Sonnenbahnen für jeden Monat im Jahr hinterlegt sind. Dies ist im Package «sungeodata» lokalisiert, hier befinden sich auch die Reader- und Parser-Klasse dazu. Der Simulator kann sich von der Klasse «SunGeoData» mit Angabe der Modulwinkel und des Monats nun den Einstrahlungsfaktor für den gegebenen Zeitpunkt holen. Die «SunGeoData»-Klasse beschafft sich die benötigten Sonnenwinkel und führt die Berechnung des Faktors durch.

9.2.2.2 *Simulation mit Sonneneinstrahlungen vom Webservice*

Vom Webservice des Instituts für Umwelt und Nachhaltigkeit der Europäischen Union können wir mittels eines HTTP⁶-Post Requests⁷ unsere Anlage-Daten übergeben und erhalten die Generator-Einstrahlung als Resultat zurück. Der Webservice übernimmt also einen grossen Teil der Berechnungen, und da dieser von einer Forschungseinrichtung der EU stammt gehen wir von einer Korrektheit der Daten aus. Wir sprechen mit dem Http-Protokoll das folgende PHP-Skript an:

<http://re.jrc.ec.europa.eu/pvgis/apps4/DRcalc.php>

⁶ Hypertext Transfer Protokoll, ein Protokoll zur Datenübertragung.

⁷ Daten-Anfrage

Die zurückgegebenen Daten werden als Text eingelesen und sehen beispielsweise wie folgt aus:

Inclination of plane:	10	deg.	
Orientation (azimuth) of plane:	0	deg.	
Time	G	Gd	Gc
08:07	28	23	47
08:22	42	32	76
08:37	57	42	117
..
14:52	84	56	188
15:07	46	46	44
15:22	39	39	38
15:37	31	31	30
15:52	23	23	22
16:07	14	14	13

G: Global irradiance on a fixed plane (W/m²)

Gd: Diffuse irradiance on a fixed plane (W/m²)

Gc: Global clear-sky irradiance on a fixed plane (W/m²)

PVGIS (c) European Communities, 2001-2012

Die Daten liegen für einen Tag viertelstündlich vor. Wir „parsen“⁸ diese mit Tabulatoren formatierte Tabelle nun in unsere Klassen zur weiteren Verarbeitung.

Die entsprechenden Klassen sind im Package «irradiationdata» enthalten.

Der Simulator fragt dabei nach einem Daten-Element für den gewünschten Tages-Zeitpunkt. Die Kommunikation sowie das Caching⁹ übernehmen die «IrradiationData» Klassen. Die Ergebnisse des Webservices liegen als Tagesverlauf für einen Durchschnittstag des gegebenen Monats vor.

Die Daten liegen dabei in einem 15-Minuten Takt vor. Wenn nun der Simulator nach der Sonneneinstrahlung zu einem spezifischen Zeitpunkt fragt, liefert die «IrradiationData» Klasse das zeitnaheste Element zurück.

⁸ Umwandlung einer Eingabe in ein für die Weiterverarbeitung geeigneteres Format.

⁹ Zwischenspeichern von geladenen Daten, damit sie nicht zu oft neu geladen werden müssen.

9.2.3 Webservices

Für die automatische Ermittlung der Sonneneinstrahlungs-Daten sind wir auf frei verfügbare Web-Services angewiesen. Dies sind im Grunde genommen über das Internet verfügbare Software-Skripte welche sich über das http-Protokoll ansprechen lassen.

In unserem Java-Package «httpconnection» haben wir für die beiden Http-Anfragetypen zwei Klassen erstellt, welche die Kommunikation über Http ein Stück weiter abstrahieren. Eine Klasse ist für den GET-Request zuständig und die andere Klasse übernimmt den POST-Request. Die Klassen sind beide noch nicht spezifisch, sie stellen lediglich ein bequemer Interface für die Http-Kommunikation zur Verfügung.

Wir benutzen zwei verschiedene Dienste, welche im Folgenden kurz erklärt werden.

9.2.3.1 Koordinaten

Der Benutzer kann über das GUI den Standort der zu simulierenden PV-Anlage in Form einer Adresse angeben. Da für den Sonnendaten-Webdienst die Koordinaten mit Längen- sowie Breitengraden benötigt werden, benutzen wir einen Webdienst von Google. Diesem können wir die Adresse mittels „GET-Request“ übergeben und erhalten als Antwort umfangreiche Informationen zum Standort. Unter anderem sind die gewünschten Koordinaten in der Antwort enthalten, welche wahlweise als JSON¹⁰ oder XML¹¹ daherkommen. Wir haben uns für das XML-Format entschieden, damit wir die Koordinaten mit Standard-Bibliotheken aus der Antwort entnehmen können.

Die zuständigen Java-Klassen befinden sich in unserem Package «location».

9.2.3.2 Sonneneinstrahlung

Mit den vom Benutzer eingegebenen Anlage-Daten kann vom Web-Dienst des Joint Research Centers der EU die Generator-Einstrahlung abgefragt werden. Wir senden einen „POST-Request“ an das PHP¹²-Skript und erhalten die gewünschten Daten im CSV-Format zurück. Aus der Antwort lesen wir die gesuchten Werte heraus und erstellen Daten-Elemente im Form von Objekten, mit welchen sich nun weiterarbeiten lässt.

¹⁰ Javascript Object Notation, Daten die in „Objekte“ eingeteilt gespeichert werden.

¹¹ Extensible Markup Language, Darstellung hierarchisch strukturierter Daten in Form von Textdateien

¹² Hypertext Preprocessor, eine Skriptsprache welche hauptsächlich zur Erstellung dynamischer Webanwendungen verwendet wird

9.3 Zusätzliche Features

9.3.1 Speichern und Laden

Damit der Benutzer die eingegebenen Daten als Projekt-Datei abspeichern und bei Bedarf wieder öffnen kann, haben wir uns entschieden dieses Feature zu implementieren. Die einfachste Methode ist, ein Objekt zu serialisieren und in eine Datei zu schreiben. Wir machen uns dafür das «Serializable» Interface der Java-Standardbibliothek zunutze. Alle zu speichernden Klassen müssen dieses Interface implementieren, wobei eigentlich gar keine Methode implementiert werden muss. Das Interface dient lediglich zur Identifizierung. Zum Abspeichern werden nun alle Objekte, welche Nutzerdaten enthalten gesammelt und in eine Datei geschrieben. Der Benutzer wählt über einen Dialog den Ordner und Dateinamen.

9.3.2 PDF-Export

Damit sich die Simulationsergebnisse präsentieren lassen und der Benutzer einen Projektbericht erstellen kann, haben wir die Funktion zum PDF-Export implementiert. Dabei werden Anlagedaten sowie Simulationsergebnisse in Diagramm-Form in eine PDF-Datei geschrieben. Die erstellte Datei lässt sich über einen Dialog benennen und abspeichern. Im Anhang zeigen wir einen Beispiel-Projektbericht, welcher aus der Applikation erstellt wurde.

9.3.3 CSV-Export

Die Berechnungsergebnisse der Simulation lassen sich über den entsprechenden Menüpunkt als CSV¹³-Dateien exportieren. Beim Export-Dialog kann der Benutzer den Ordner im Dateisystem wählen, in welchen die Dateien abgelegt werden sollen. Danach speichert die Applikation die Simulationsergebnisse über den Zeitraum einer Woche, eines Jahres sowie das Jahrestotal. Die CSV-Dateien lassen sich in Excel öffnen und über den entsprechenden Menüpunkt werden die Werte auf mehrere Spalten verteilt. Im Anhang findet sich ein Beispiel eines solchen Exports.

¹³ CSV = Comma Separated Values (Formatierungsart)

10 Wirtschaftlichkeit

10.1 Wirtschaftlichkeit der Applikation

Zur Einschätzung der Wirtschaftlichkeit dieses Software-Projekts rechnen wir anhand der geplanten Arbeitsdauer jedes Pakets die entstehenden Kosten aus. Aus diesen können wir die Gesamtkosten des Projekts bestimmen. Als Stundenansatz haben wir jeweils **CHF 100 / Std.** eingesetzt, um alle anfallenden Kosten decken zu können. Die folgende Tabelle soll die Berechnung der Projektkosten übersichtlich aufzeigen:

Arbeitspaket	Anzahl Stunden	Kosten
Dokumentation	41 Std.	CHF 4100.-
Recherche	13 Std.	CHF 1300.-
Programmieren	83 Std.	CHF 8300.-
Präsentation	7 Std.	CHF 700.-
Total	144 Std.	CHF 14'400.-

Aus den voraussichtlichen Gesamtkosten des Projekts möchten wir nun die Verkaufspreise anhand verschiedener Stückzahlen bestimmen. Das Produkt würde dabei mit Einzel-Lizenzen verkauft werden, es wäre jedoch auch eine erweiterte Lizenz mit inbegriffenem technischem Support und Beratung denkbar.

Wir haben aufgrund von uns festgelegten Verkaufspreisen die nötigen Mindest-Verkaufsmengen berechnet:

Total Kosten	Verkaufspreis/Lizenz	Mindest-Menge
CHF 14'400.-	CHF 25.0	576 Stk.
	CHF 50.0	288 Stk.
	CHF 100.0	144 Stk.
	CHF 150.0	96 Stk.

Mit der verkauften Mindest-Menge (Break-Even-Menge) sind unsere Kosten gedeckt, alle weiteren Verkäufe des Produkts generieren den Gewinn.

Da es sich um eine professionell eingesetzte Software handelt, für dessen Entwicklung es einiges an Fachkenntnissen bedarf, kann der Preis auch entsprechend höher angelegt sein. Wir denken an ein Preissegment zwischen CHF 90.- bis CHF 110.-. Die dafür erforderliche Verkaufsmenge lässt sich bereits in der Schweiz und Deutschland sicherlich gut erreichen. Ausserdem ist die Software damit auch für Privatpersonen erschwinglich.

Für einen internationalen Verkauf müsste die Software noch in weitere Sprachen übersetzt werden, was sich aufgrund der Kosten wiederum auf den Verkaufspreis niederschlagen würde.

10.2 Wirtschaftlichkeit einer Photovoltaik-Anlage analysieren

Die Wirtschaftlichkeit einer Photovoltaik-Anlage möchten wir anhand des eingesparten sowie des verkauften Stromes bestimmen. Im Falle des eingesparten Stromes entsprechen die Kosten je Kilowatt-Stunde dem Einkaufspreis. Wenn Strom verkauft wird kommt der Verkaufspreis zum Tragen.

Anhand der Strom-Ein- sowie Verkaufsbilanz lässt sich ein Gewinn pro Zeiteinheit (zum Beispiel jährlich) berechnen. Des Weiteren lässt sich anhand diesem eine Amortisationszeit ausrechnen; dies ist die Zeit, welche benötigt wird um die kompletten Anlage-Kosten wieder einzunehmen.

Im Folgenden ermitteln wir die Amortisationszeit einer Beispiel-Anlage mit unserer Applikation als Hilfsmittel.

Anlagen-Parameter:

Anzahl Panel:	10
Grösse je Panel:	1.64 m ²
Wirkungsgrad / Panel:	0.21
Aufstellwinkel:	10°
Azimut:	0°
Batterie-Kapazität:	2 kWh

Anlage-Kosten:

Solar-Panel	CHF 250.- je Stück
Batterie	CHF 9950.-
Wechselrichter	CHF 1200.-
Planung und Installation	CHF 5000.-
Total	CHF 18'650.-

Strom-Preise:

Einkaufspreis / Kilowattstunde	CHF 0.15
Verkaufspreis / Kilowattstunde	CHF 0.20

Den Stromverbrauch haben wir als konstant 500 Watt festgelegt, als Anlagen-Standort haben wir Baden, Schweiz eingestellt. Die Sonnendaten werden somit vom Webservice heruntergeladen. Wir ermitteln nun die Kostenbilanz für jede Woche eines Monats im Jahr.

Die Kostenbilanz ergibt sich folgendermassen:

Bilanz = Verkaufsgewinn + Ersparnis

Januar	CHF 2.18	Februar	CHF 7.23
März	CHF 13.92	April	CHF 19.76
Mai	CHF 22.82	Juni	CHF 24.65
Juli	CHF 24.47	August	CHF 21.17
September	CHF 15.36	Oktober	CHF 8.38
November	CHF 3.27	Dezember	CHF 0.09

Mittelwert: $\frac{\sum \text{Kostenbilanzen}}{12} = \text{CHF } 13.61$

Bilanz / Tag: $\frac{\text{Mittelwert Woche}}{7} = \text{CHF } 1.94$

Pro Tag im Jahr wird also Durchschnittlich der Betrag von CHF 1.94 gespart.

Für ein ganzes Jahr ergibt sich somit ein Betrag von $365 * \text{CHF } 1.94 = \text{CHF } 709.58$.

Für die kompletten Anlage-Kosten von CHF 18'650.- ergibt sich somit eine Amortisationszeit von:

$$\frac{\text{Total Anlage} - \text{Kosten}}{\text{jährlicher Gewinn}} = \text{Amortisationszeit}$$

$$\frac{\text{CHF } 18'650.-}{\text{CHF } 709.58} = \mathbf{26.28 \text{ Jahre}}$$

Ein Überblick der Kostenbilanzen über die Laufzeit von 30 Jahren:

Jahr	Total Ersparnis	Übrige Kosten
1	709,58 CHF	17.940,42 CHF
5	3.547,89 CHF	15.102,11 CHF
10	7.095,77 CHF	11.554,23 CHF
15	10.643,66 CHF	8.006,34 CHF
20	14.191,55 CHF	4.458,45 CHF
25	17.739,43 CHF	910,57 CHF
30	21.287,32 CHF	- 2.637,32 CHF

11 Anhang

11.1 Verwendete Hilfsmittel

Bezeichnung:	Verwendung:
GIT www.git-scm.com	Für die Versionierung der Software und die Synchronisation der Dokumentation verwendet.
MS Project www.products.office.com/de-de/project/project-and-portfolio-management-software	Für die Projektstrukturplanung und den Projektablaufplan verwendet.
MS Visio www.products.office.com/de-ch/visio	Für UML Diagramme, Timeline und den „druckfertigen“ Projektstrukturplan verwendet.
MS Word www.products.office.com/de-ch/word	Für die Erstellung der Dokumentation verwendet.
MS Excel www.products.office.com/de-ch/excel	Zur Erstellung von Tabellen und für die Erarbeitung der PV Berechnungen verwendet.
Google Maps API www.developers.google.com/maps/?hl=de	Die Google Maps API haben wir verwendet um die Koordinaten von Adressen zu ermitteln, damit wir beim geografischen PV-Informationssdienst die Sonneinstrahlungs-Daten für die PV-Anlage bekommen.
«Photovoltaic Geographical Information System» Institut für erneuerbare Energien der europäischen Kommission re.jrc.ec.europa.eu/pvgis/apps4/pvest.php	Anhand der Koordinaten einer Anlage sowie der Anlagen-Winkel erhalten wir von dieser Datenbank den Verlauf der Sonneneinstrahlung für einen bestimmten Zeitraum.
Bitbucket www.bitbucket.org	«Git»-Server der private und geteilte «Repositories» gratis hostet.
IntelliJ Ideas Community Edition www.jetbrains.com/idea/	Wir haben für die Entwicklung der Software die IDE von JetBrains verwendet.
ITextPDF www.itextpdf.com	Java Bibliothek für PDF Exports. Damit haben wir den PDF Export in unserer Software Implementiert.
Log4J www.logging.apache.org/log4j	Haben wir verwendet, um ein Logsystem zu implementieren.
Trello Task Board www.trello.com	Haben wir verwendet, um auch die „kleinen“ Tasks zu verwalten.

11.2 Quellenangaben

Abbildung 3.1 - Organigramm erstellt mit Microsoft Visio 2013

© Shane Hofstetter & Simon Müller

Abbildung 3.2 - Strukturplan, erstellt mit Microsoft Visio 2013

© Shane Hofstetter & Simon Müller

Abbildung 3.3 - Ablaufplan, erstellt mit Microsoft Project 2013

© Shane Hofstetter & Simon Müller

Abbildung 3.4 - Terminplan, erstellt mit MS Visio 2013

© Shane Hofstetter & Simon Müller

Abbildung 4.1 - Strommix Schweiz

Quelle: Bundesamt für Energie (BFE) - www.strom.ch/de/energie/energiefakten/produktion-und-strommix.html

Abbildung 4.2 - Montage einer Photovoltaik Aufdach-Anlage

Quelle: www.solaranlage.org/files/2011/09/pv_montage.jpg

Abbildung 6.1 - JavaFX Icon

Quelle: www.java.com/ga/images/javafx_logo.gif

Abbildung 6.2 - Microsoft WPF Icon.

Quelle:

www.blogs.msdn.com/blogfiles/tims/WindowsLiveWriter/IntroducingtheThirdMajorReleaseofWindows_12A7F/WPF_Logo_2.png

Abbildung 7.1 - Eine mit der JFreeChart Bibliothek generierte Line-Chart

Quelle: www.jfree.org/jfreechart/images/XYSplineRendererDemo1a_254.png

Abbildung 7.2 - Eine mit der JavaFX generierte Line-Chart

Quelle: www.docs.oracle.com/javase/8/javafx/user-interface-tutorial/img/line-sample.png

Abbildung 8.1 - Entwurf UML-Diagramm für den Aufbau der Software

© Shane Hofstetter & Simon Müller

Abbildung 8.2 - Zusammensetzung des Modulwinkels Quelle: Simulation der Abschattungsverluste bei solarelektrischen Systemen (1996) - Volker Quasching [Kap 3.1]

Abbildung 8.3 - Berechnung des Generator-Winkels Quelle: Simulation der Abschattungsverluste bei solarelektrischen Systemen (1996) - Volker Quasching [Kap 3.1]

Abbildung 8.4 - Sonnenwinkel-Jahreszeit Quelle: Simulation der Abschattungsverluste bei solarelektrischen Systemen (1996) - Volker Quasching [Kap 3.1]

Abbildung 8.5 - Direkte Generator-Einstrahlungsenergie Quelle: Simulation der Abschattungsverluste bei solarelektrischen Systemen (1996) - Volker Quasching [Kap 3.1]

Abbildung 8.6 - Entwurfszeichnungen erstellt mit der Software «Evolus Pencil»

© Shane Hofstetter & Simon Müller

Abbildung 9.1 - Screenshot der Applikation

© Shane Hofstetter & Simon Müller

Abbildung 9.2 - Einstellungen zum Photovoltaik-Panel

© Shane Hofstetter & Simon Müller

Abbildung 9.3 - Einstellungen zur Batterie

© Shane Hofstetter & Simon Müller

Abbildung 9.4 - Simulationseinstellungen

© Shane Hofstetter & Simon Müller

Abbildung 9.5 - Einstellungen Strompreise

© Shane Hofstetter & Simon Müller

Abbildung 9.6 - Einstellungen Anlagekosten

© Shane Hofstetter & Simon Müller

Abbildung 9.7 - Erweiterte Anlagenparameter

© Shane Hofstetter & Simon Müller

Abbildung 9.8 - Solarpanel-Datenbank

© Shane Hofstetter & Simon Müller

Abbildung 9.9 - Tagesverlauf der Simulations-Ergebnisse

© Shane Hofstetter & Simon Müller

Abbildung 9.10 - Wochenverlauf der Simulations-Ergebnisse

© Shane Hofstetter & Simon Müller

Abbildung 9.11 - Wochentotal

© Shane Hofstetter & Simon Müller

Abbildung 9.12 - Jahresverlauf

© Shane Hofstetter & Simon Müller

Abbildung 9.13 - Kumulierte Ergebnisse des gesamten Jahres

© Shane Hofstetter & Simon Müller

Abbildung 9.14 - Kumulierte Gewinne und Kosten des gesamten Jahres

© Shane Hofstetter & Simon Müller

Abbildung 9.15 - Amortisationszeit

© Shane Hofstetter & Simon Müller

Abbildung 9.16 - Ausschnitt eines Screenshots der Applikation

© Shane Hofstetter & Simon Müller

Abbildung 9.17 - Interaktives Liniendiagramm zum Eingeben des Stromverbrauchs

© Shane Hofstetter & Simon Müller

11.3 Angehängte Dokumente

- **Arbeitsjournal**
- **UML Diagramm**
- **Programm-Exporte**
 - **Projektbericht als PDF**
 - **Simulationsergebnisse als CSV**
- **Gantt-Chart**

Economy

- + getAvailableCurrencies() : Set<Currency>
- + getCurrency() : Currency
- + setCurrency(currency : Currency)
- + getCurrencyCode() : String
- + setCurrencyByCode(currencyCode : String)
- + addListener(listener : CurrencyChangeListener)



<<Schnittstelle>>

CurrencyChangeListener

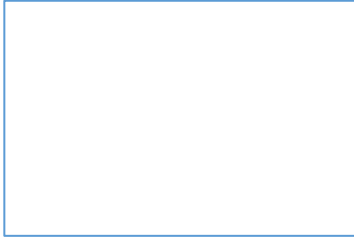
- + changed(currency : Currency)

PowerPrice

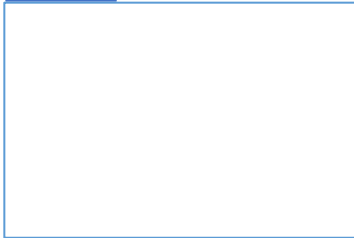
- + init()
- + getPowerUnit() : String
- + getPurchaseKwhPrice() : BigDecimal
- + setPurchaseKwhPrice(purchaseKwhPrice : BigDecimal)
- + getSellingKwhPrice() : BigDecimal
- + setSellingKwhPrice(sellingKwhPrice : BigDecimal)
- + getPurchaseCosts(kwhAmountPurchased : double) : BigDecimal
- + getSalesProfit(kwhAmountSold : double) : BigDecimal

gui.components

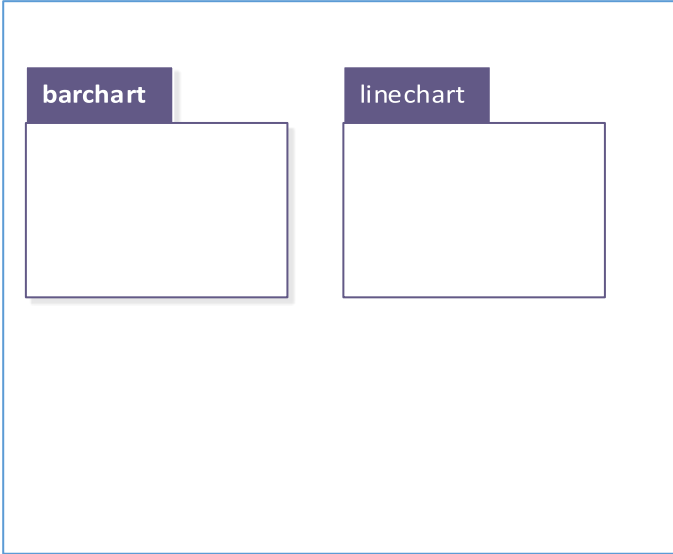
controllers



views



pvcharts



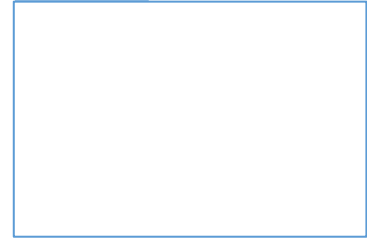
barchart



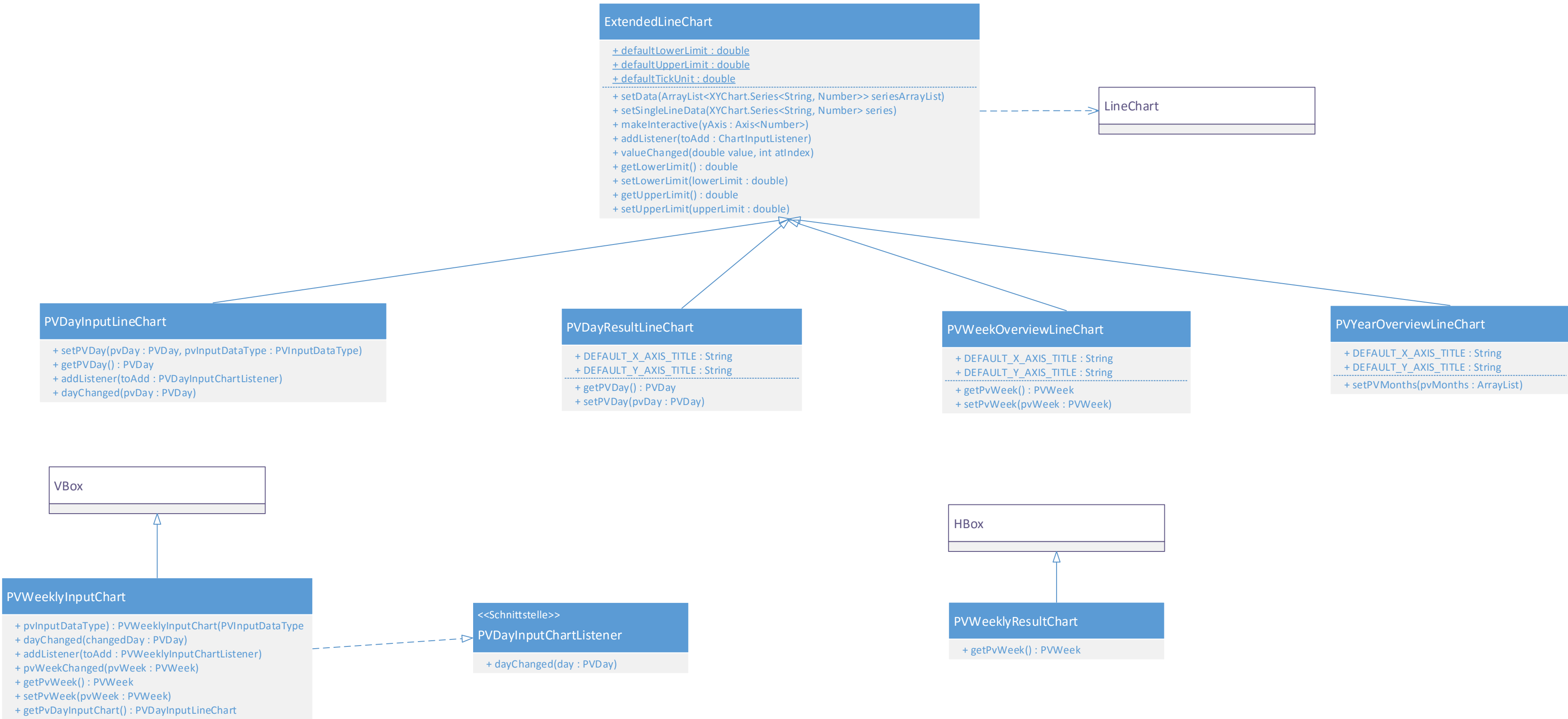
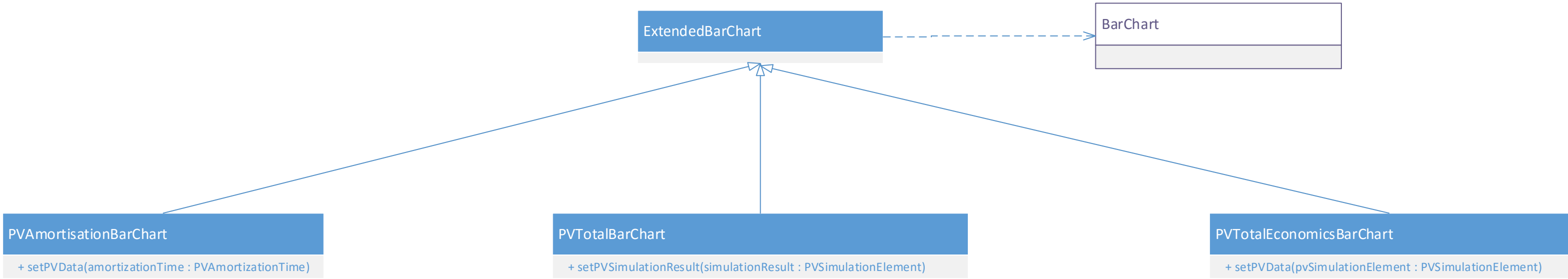
linechart

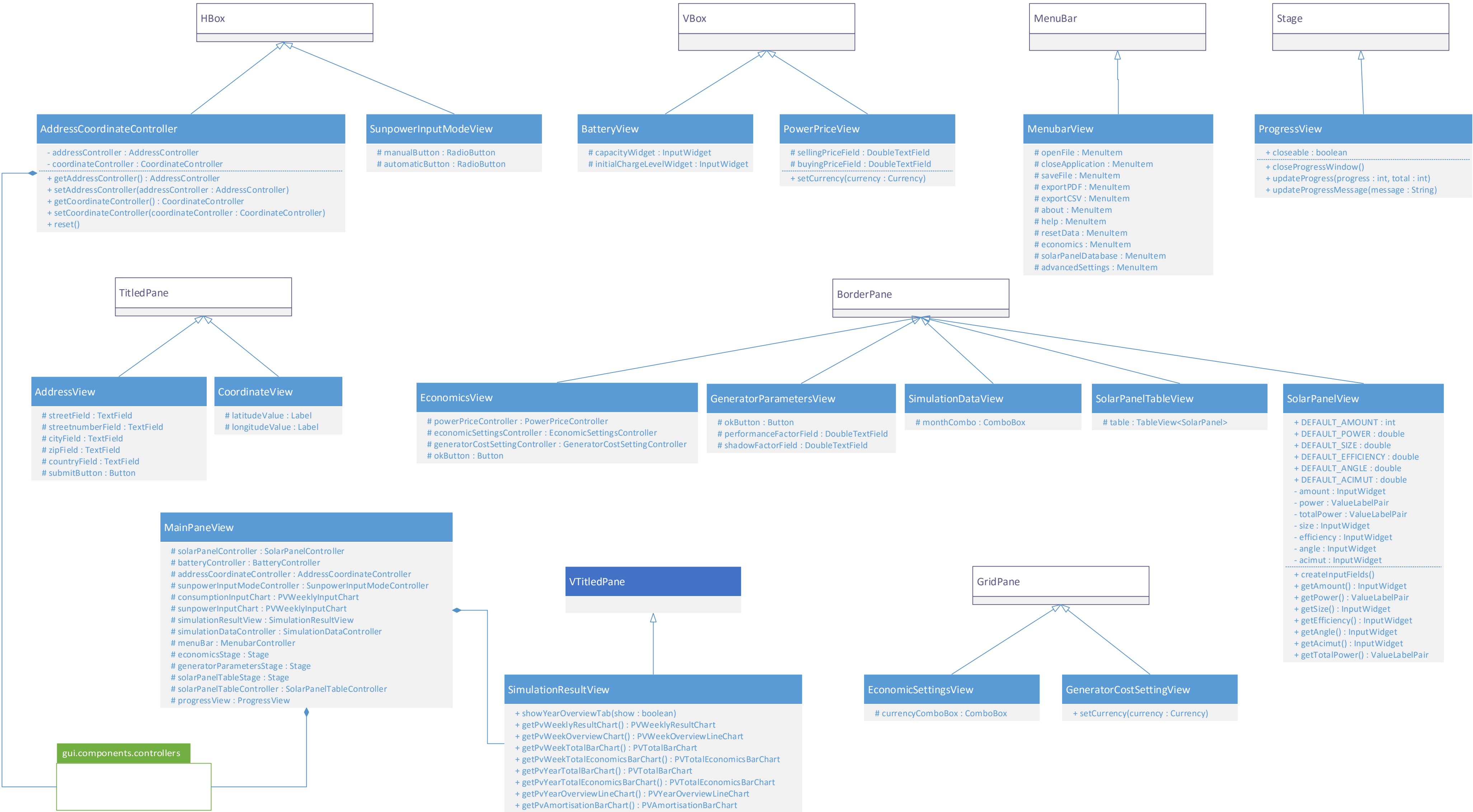


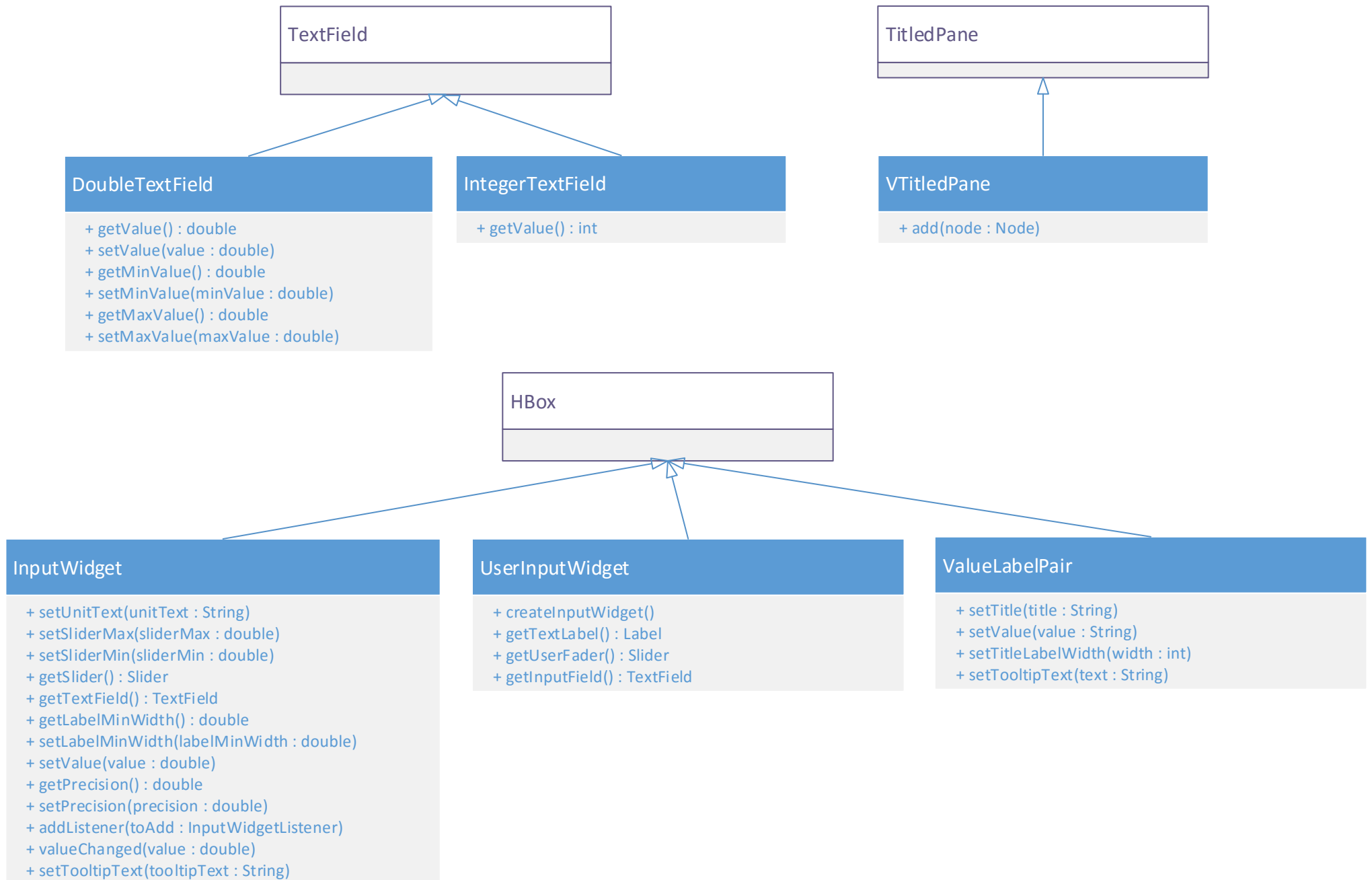
widgets



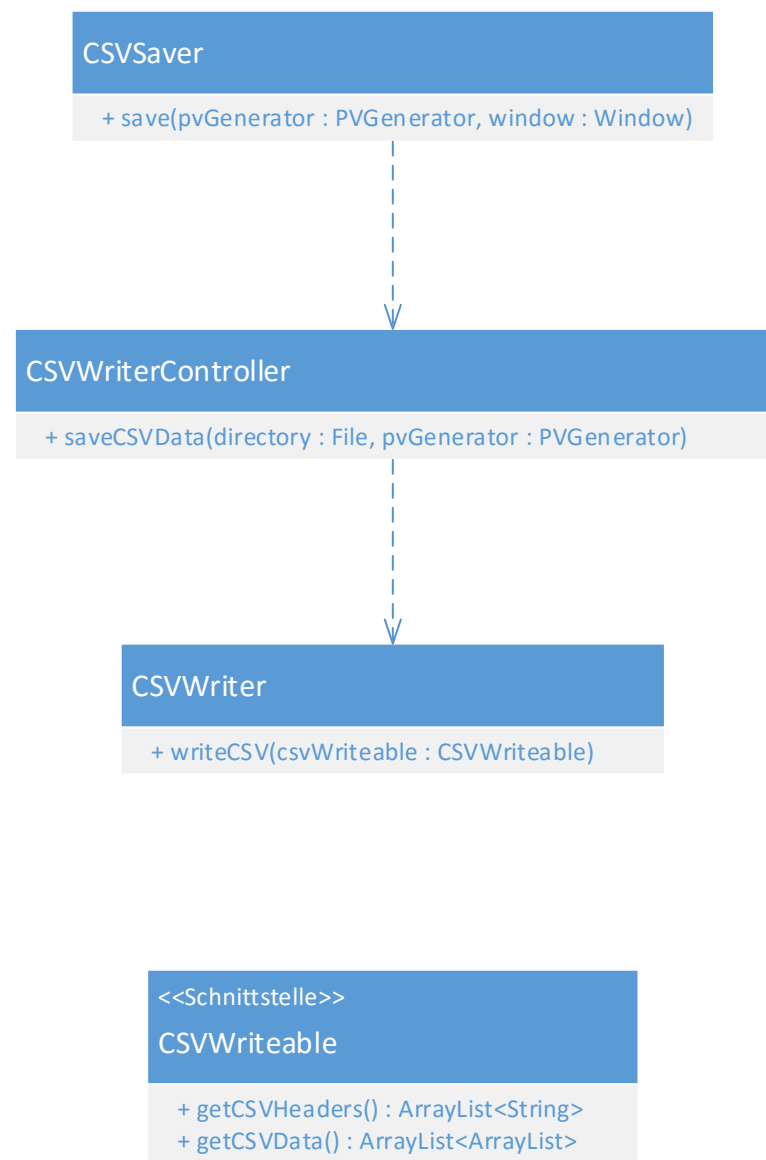




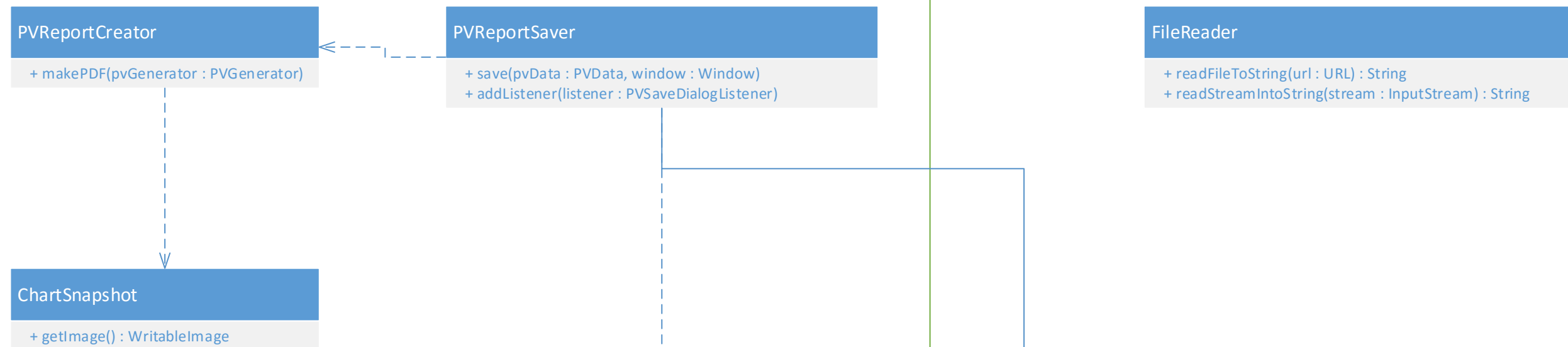




csv



pdfexport



pvdata



IrradiationData

+ getIrradiationDataElements() : ArrayList<IrradiationDataElement>

+ setIrradiationDataElements(irradiationDataElements : ArrayList)

+ getWeeklyIrradiation() : ArrayList<Double>

+ getRealRadiationWhForTimeOfDay(irradiationDataElements : ArrayList, time : long) : double

+ getRealRadiationWhForTimeOfDay(time : long) : double

IrradiationDataElement

+ setTimeString(timeString : String)

+ getTime() : Date

+ setTime(time : Date)

+ getGlobalIrradiance() : double

+ setGlobalIrradiance(globalIrradiance : double)

+ getDiffuseIrradiance() : double

+ setDiffuseIrradiance(diffuseIrradiance : double)

+ getClearSkyIrradiance() : double

+ setClearSkyIrradiance(clearSkyIrradiance : double)

IrradiationDataElements

+ getMonth() : Month

+ setMonth(month : Month)

+ getIrradiationDataElements() : ArrayList<IrradiationDataElement>

+ setIrradiationDataElements(irradiationDataElements : ArrayList)

IrradiationDataParser

+ parseCSVData(csv : String) : ArrayList<IrradiationDataElement>

IrradiationDataFetcher

-Mitgliedsname

+ getIrradiationDataElements(month : Month, solarPanelField : SolarPanelField, coordinates : Coordinates) : ArrayList<IrradiationDataElement>

IrradiationDataRequest

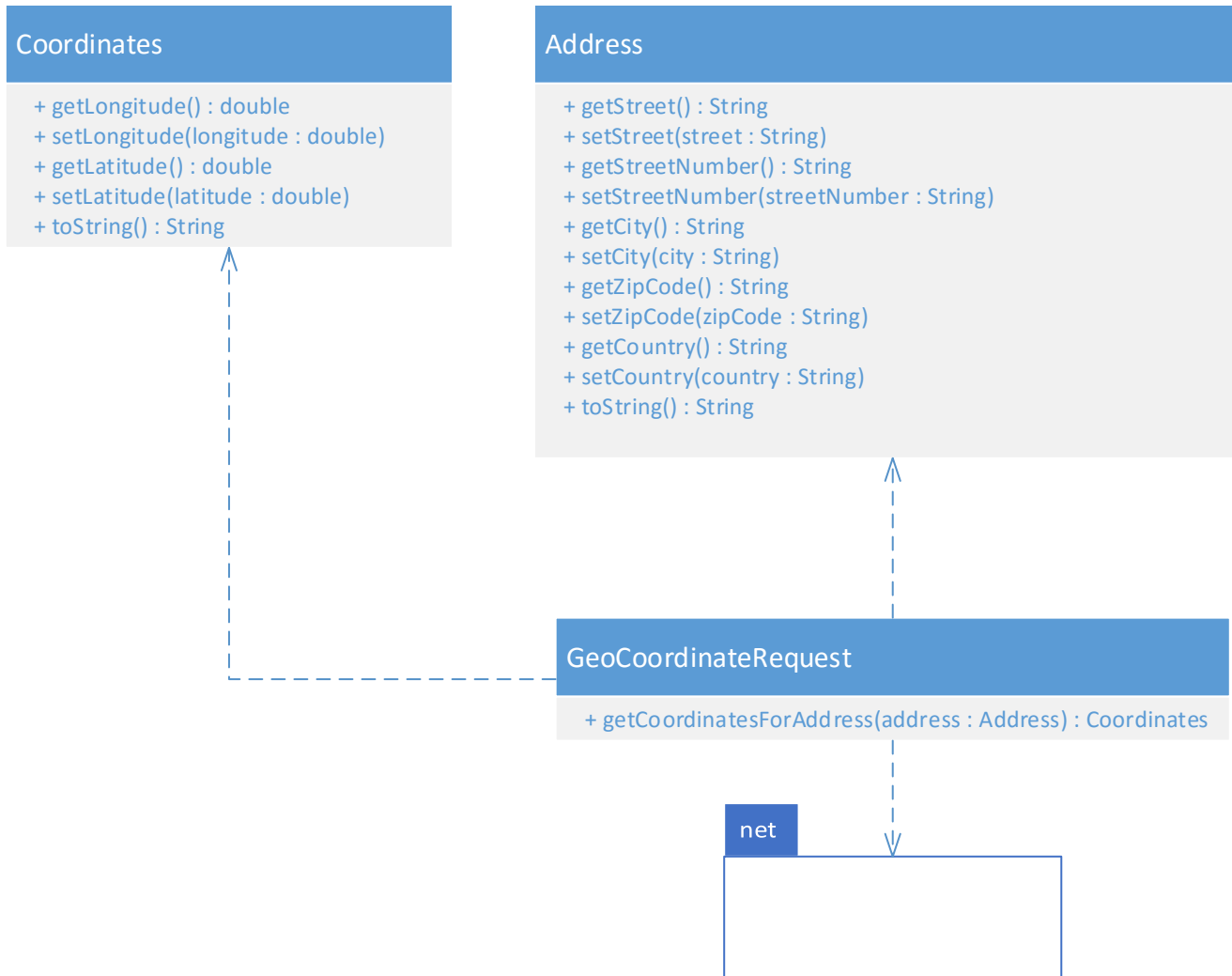
+ makeRequest(month : Month, solarPanelField : SolarPanelField, coordinates : Coordinates) : String

+ makeRequest(month : Month, panelAngle : double, panelAzimut : double, coordinates : Coordinates) : String

+ makeRequest(month : Month, panelAngle : double, panelAzimut : double, latitude : double, longitude : double) : String

net





HttpGetRequest

+ sendGETRequest(url : String) : String

HttpPostRequest

+ sendPOSTRequest(url : String, urlParameters : String) : String

SolarPanelReader

+ readModules() : ArrayList<SolarPanel>

SolarPanelDatabase

+ load()
+ getSolarPanels() : ObservableList<SolarPanel>

SolarPanelField

+ getAmount() : int
+ setAmount(amount : int)
+ getVerticalAngle() : double
+ setVerticalAngle(verticalAngle : double)
+ getAzimut() : double
+ setAzimut(azimut : double)
+ getSolarPanel() : SolarPanel
+ setSolarPanel(solarPanel : SolarPanel)
+ getInstalledPower() : double
+ getTotalSize() : double
+ getTotalCost() : BigDecimal

SolarPanel

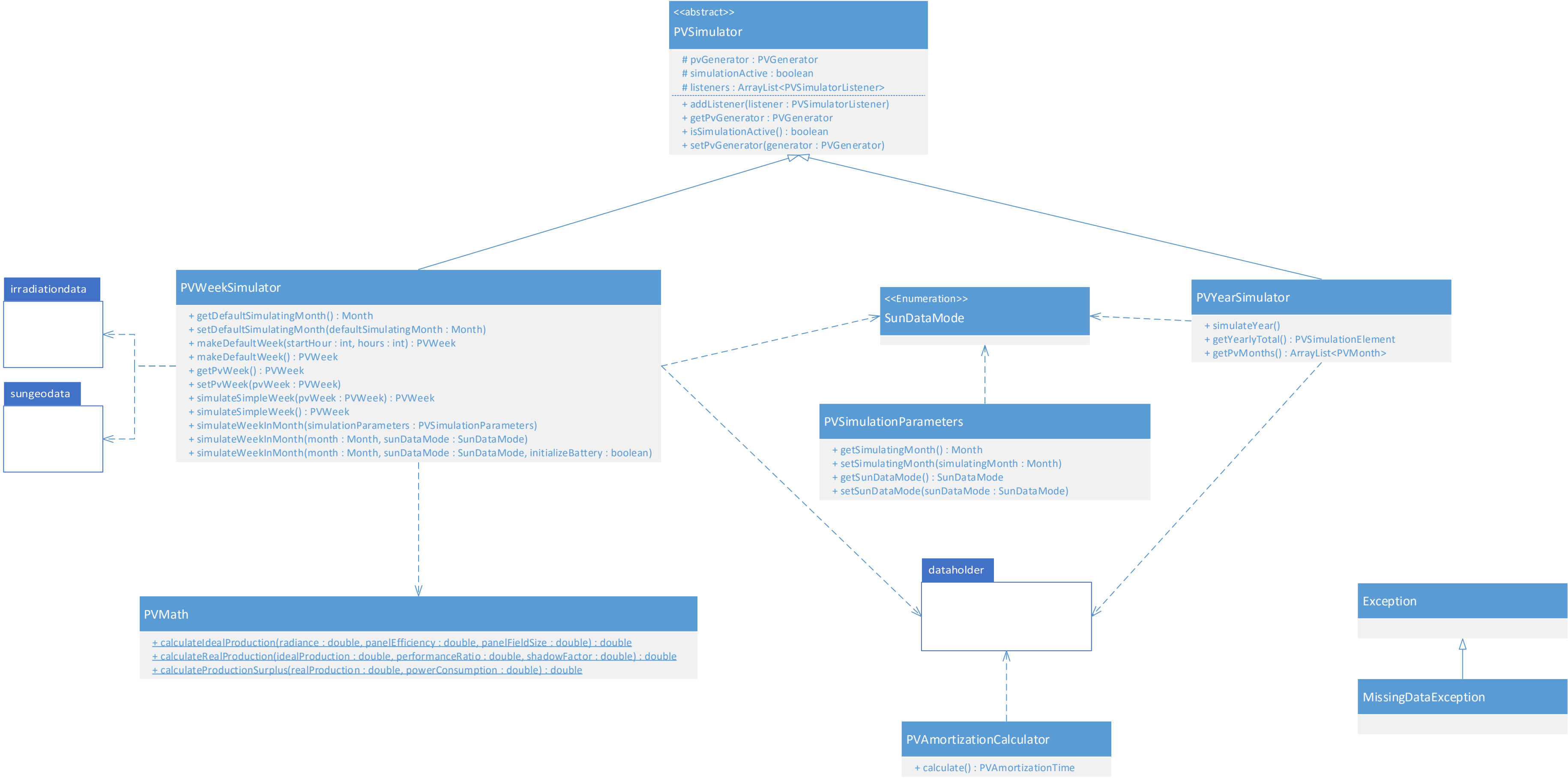
+ getHeightMeters() : double
+ setHeightMeters(heightMeters : double)
+ getWidthMeters() : double
+ setWidthMeters(widthMeters : double)
+ getEfficiency() : double
+ setEfficiency(eficiency : double)
+ getPower() : double
+ setPower(power : double)
+ getManufacturer() : String
+ setManufacturer(manufacturer : String)
+ getTypeName() : String
+ setTypeName(typeName : String)
+ getDetails() : String
+ setDetails(details : String)
+ getCost() : BigDecimal
+ setCost(cost : BigDecimal)
+ getSize() : double
+ setSize(size : double)
+ setSizeByPowerAndEfficiency()
+ calculateEfficiency() : double
+ calculatePower() : double
+ getEfficiencyPercentage() : double

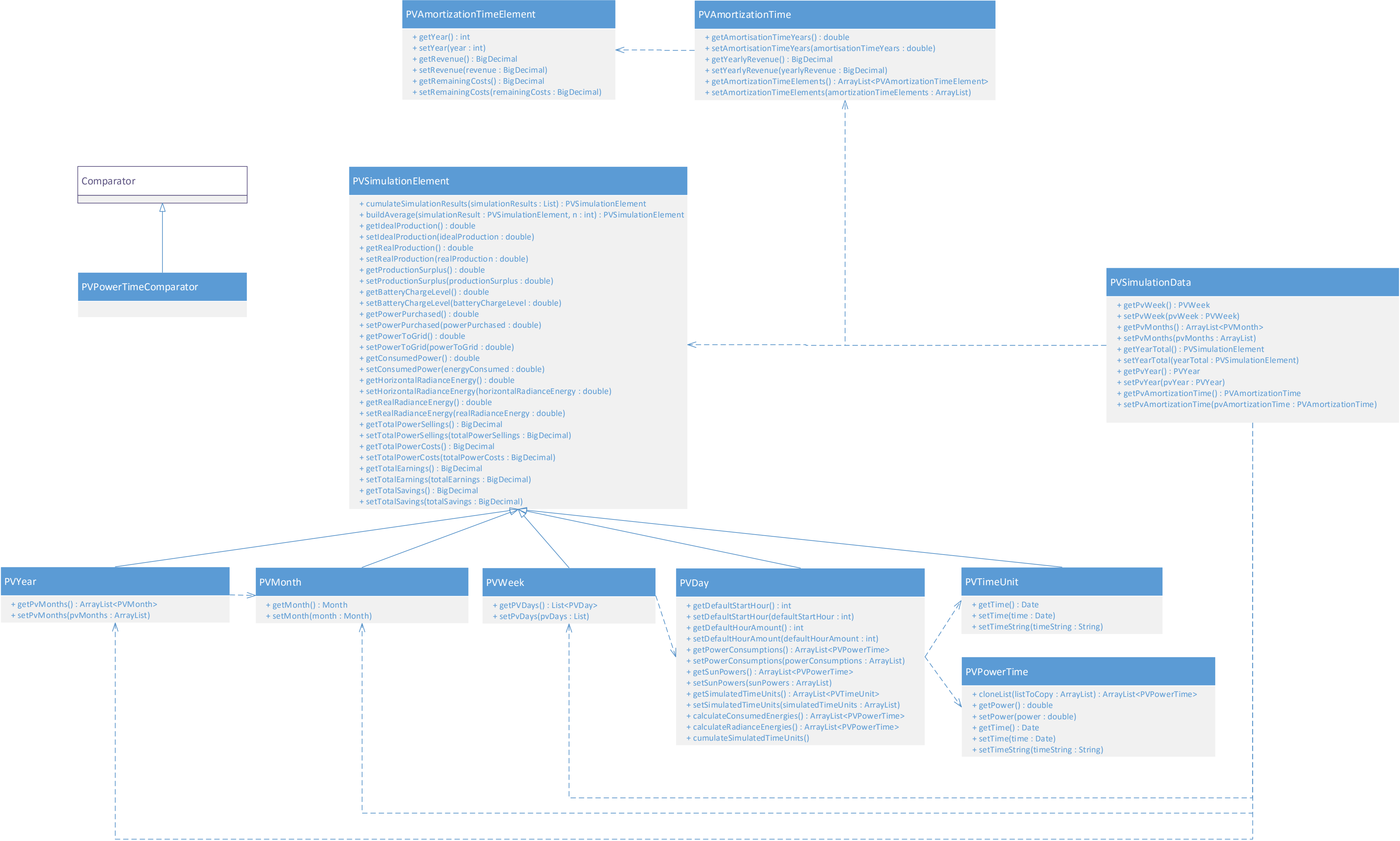
PVGenerator

+ getPerformanceRatio() : double
+ setPerformanceRatio(performanceRatio : double)
+ getSolarPanelField() : SolarPanelField
+ setSolarPanelField(solarPanelField : SolarPanelField)
+ getBattery() : Battery
+ setBattery(battery : Battery)
+ getShadowFactor() : double
+ setShadowFactor(shadowFactor : double)
+ getCoordinates() : Coordinates
+ setCoordinates(coordinates : Coordinates)
+ getAddress() : Address
+ setAddress(address : Address)
+ getPowerPrice() : PowerPrice
+ setPowerPrice(powerPrice : PowerPrice)
+ getInverter() : Inverter
+ setInverter(inverter : Inverter)
+ getTotalCosts() : BigDecimal
+ getPlanningAndInstallationCosts() : BigDecimal
+ setPlanningAndInstallationCosts(planningAndInstallationCosts : BigDecimal)
+ getSimulationParameters() : PVSimulationParameters
+ setSimulationParameters(simulationParameters : PVSimulationParameters)
+ getSimulationData() : PVSimulationData
+ setSimulationData(simulationData : PVSimulationData)

Battery

+ getCapacityWh() : double
+ setCapacityWh(capacityWh : double)
+ getCapacitykWh() : double
+ getManufacturer() : String
+ setManufacturer(manufacturer : String)
+ getTypeName() : String
+ setTypeName(typeName : String)
+ getDetails() : String
+ setDetails(details : String)
+ getChargeLevel() : double
+ charge(amount : double) : double
+ discharge(amount : double) : double
+ getChargeCoefficient() : double
+ setChargeCoefficient(chargeCoefficient : double)
+ getInitialChargeLevel() : double
+ setInitialChargeLevelWh(initialChargeLevel : double)
+ setInitialChargeLevelkWh(initialChargeLevelkWh : double)
+ getCost() : BigDecimal
+ setCost(cost : BigDecimal)
+ initialize()





SunGeoData

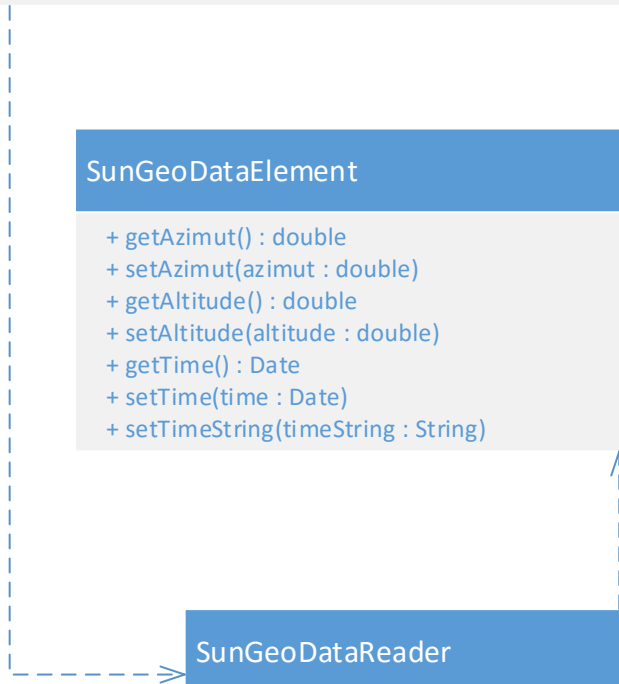
```
+ getGlobalRadianceFactorForAverageDay(month : Month, solarPanelField : SolarPanelField) : double  
+ getGlobalRadianceFactorForAverageDay(month : Month, panelAngle : double, panelAzimut : double) : double  
+ getRadianceFactorForTimeOfDay(month : Month, time : long, panelAngle : double, panelAzimut : double) : double
```

SunGeoDataElement

```
+ getAzimut() : double  
+ setAzimut(azimut : double)  
+ getAltitude() : double  
+ setAltitude(altitude : double)  
+ getTime() : Date  
+ setTime(time : Date)  
+ setTimeString(timeString : String)
```

SunGeoDataReader

```
+ readSunGeoData(month : Month) : ArrayList<SunGeoDataElement>
```



DateTimeUtil

- [+ getHoursFromTime\(time : long\) : double](#)
- [+ subtractMinutesFromTime\(time : long, minutes : int\) : long](#)
- [+ addMinutesToTime\(time : long, minutes : int\) : long](#)
- [+ formatTime\(time : long\) : String](#)
- [+ getNumberOfDaysInMonth\(month : Month\) : int](#)

Utilities

- [+ limitValue\(value : double, lowerLimit : double, upperLimit : double\) : double](#)
- [+ round\(value : double, decimals : int\) : double](#)
- [+ countMatches\(str : String, sub : String\) : int](#)



Projektbericht

Netzgekoppelte Photovoltaik-Anlage mit Eigenverbrauch

1. Photovoltaik-Anlagenparameter

1.1. PV-Panel

Anzahl	10 Stk.
Leistung	344.4 W
Grösse	1.64 m ²
Wirkungsgrad	21.0%
Aufstellwinkel	10.0°
Azimut	0.0°

1.2. Batterie

Kapazität	2.0 kWh
-----------	---------

2. Kosten

2.1. Strompreise

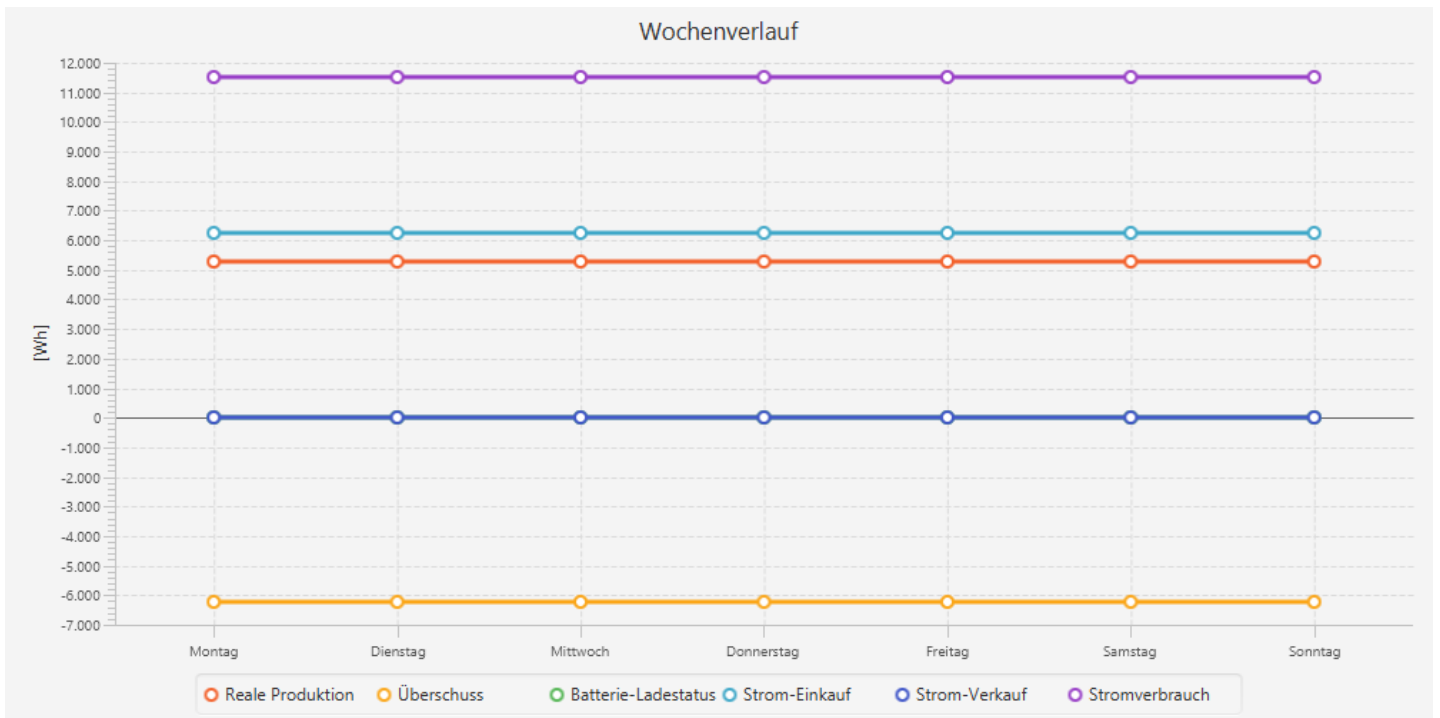
Einkaufspreis / kWh	0.15 CHF/kWh
Verkaufspreis / kWh	0.2 CHF/kWh

2.2. Projektkosten

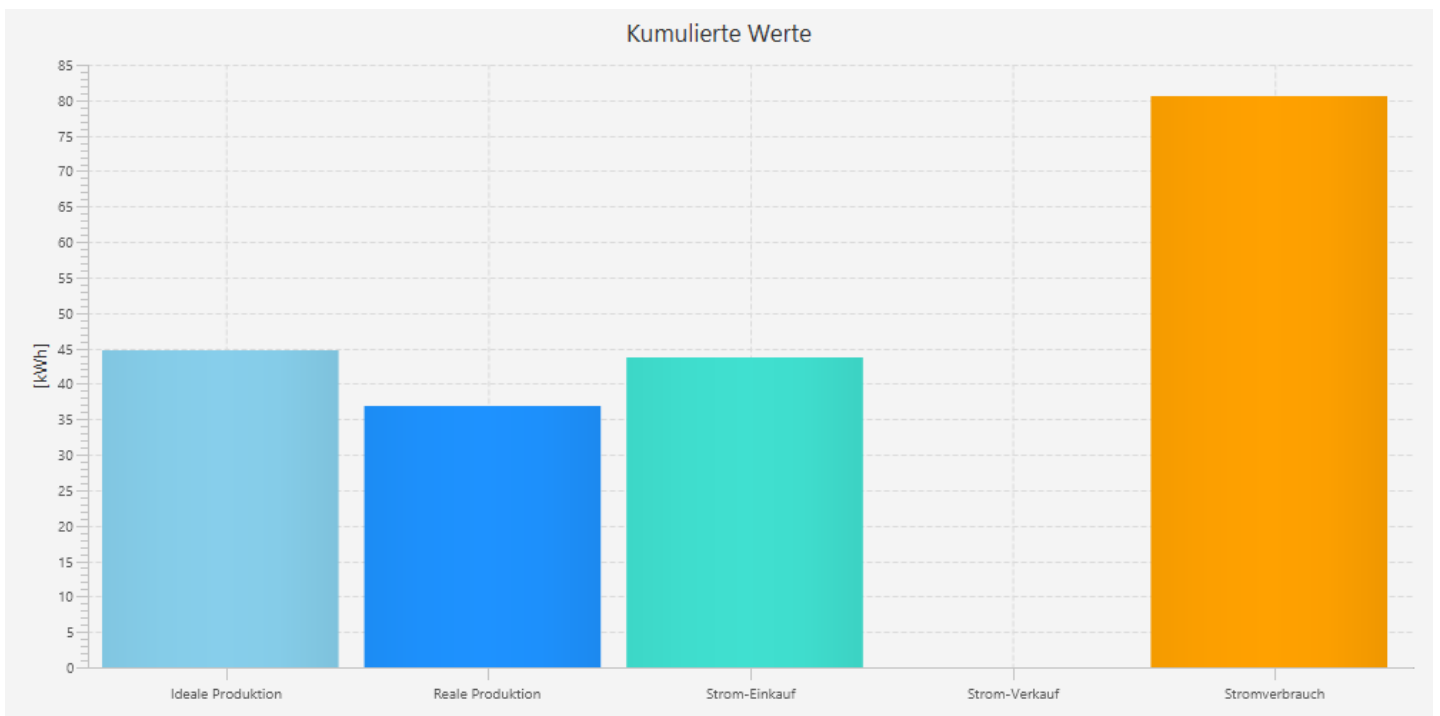
Photovoltaikpanel Stk.	250.0 CHF
Batterie	2000.0 CHF
Wechselrichter	2000.0 CHF
Planung und Installation	4000.0 CHF

3. Simulationsergebnisse

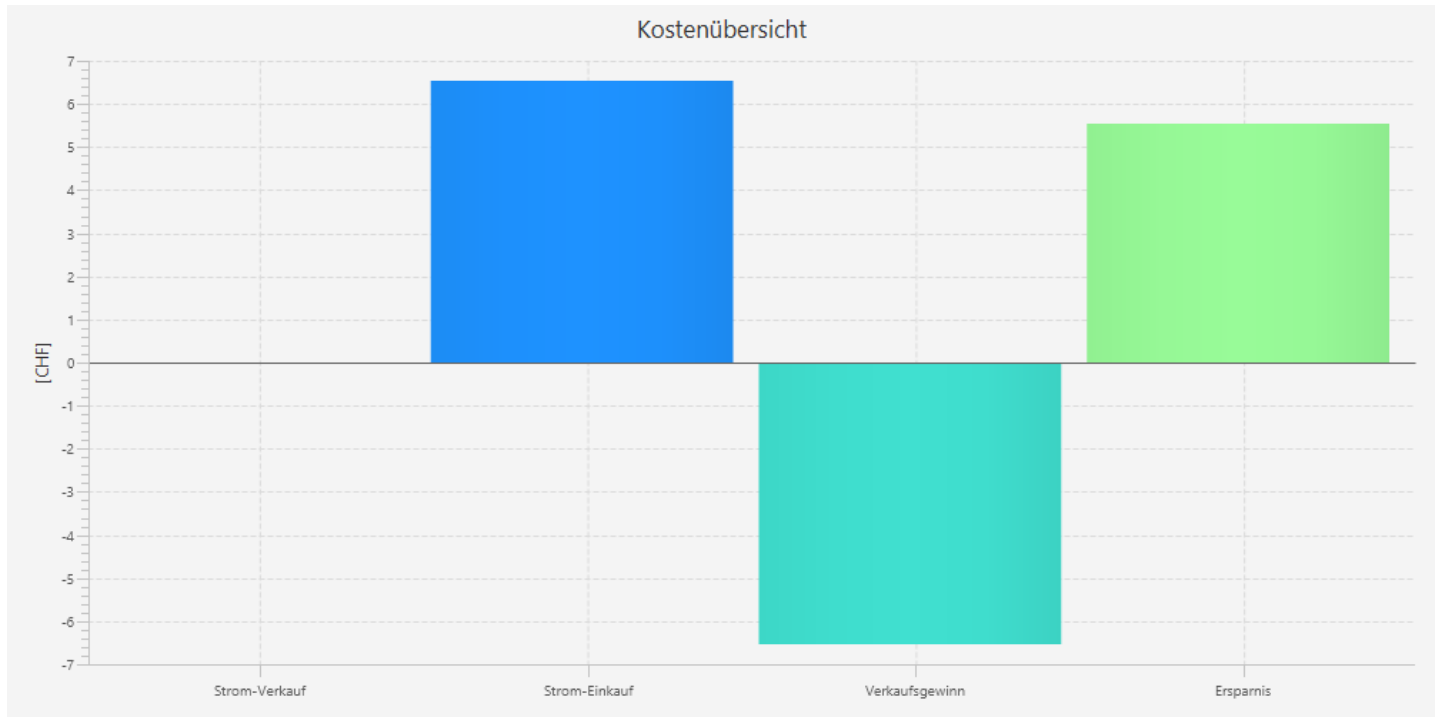
3.1. Wochenverlauf im Januar



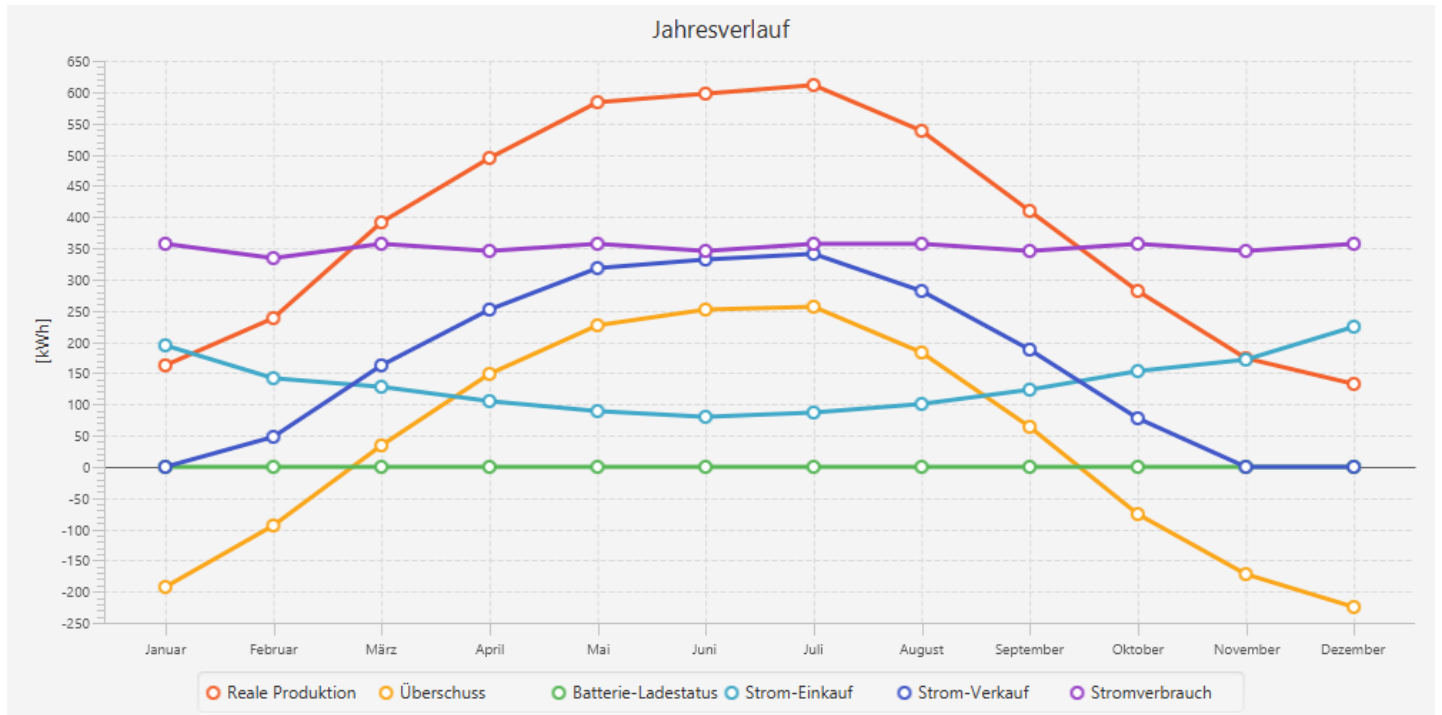
3.2. Wochenübersicht im Januar



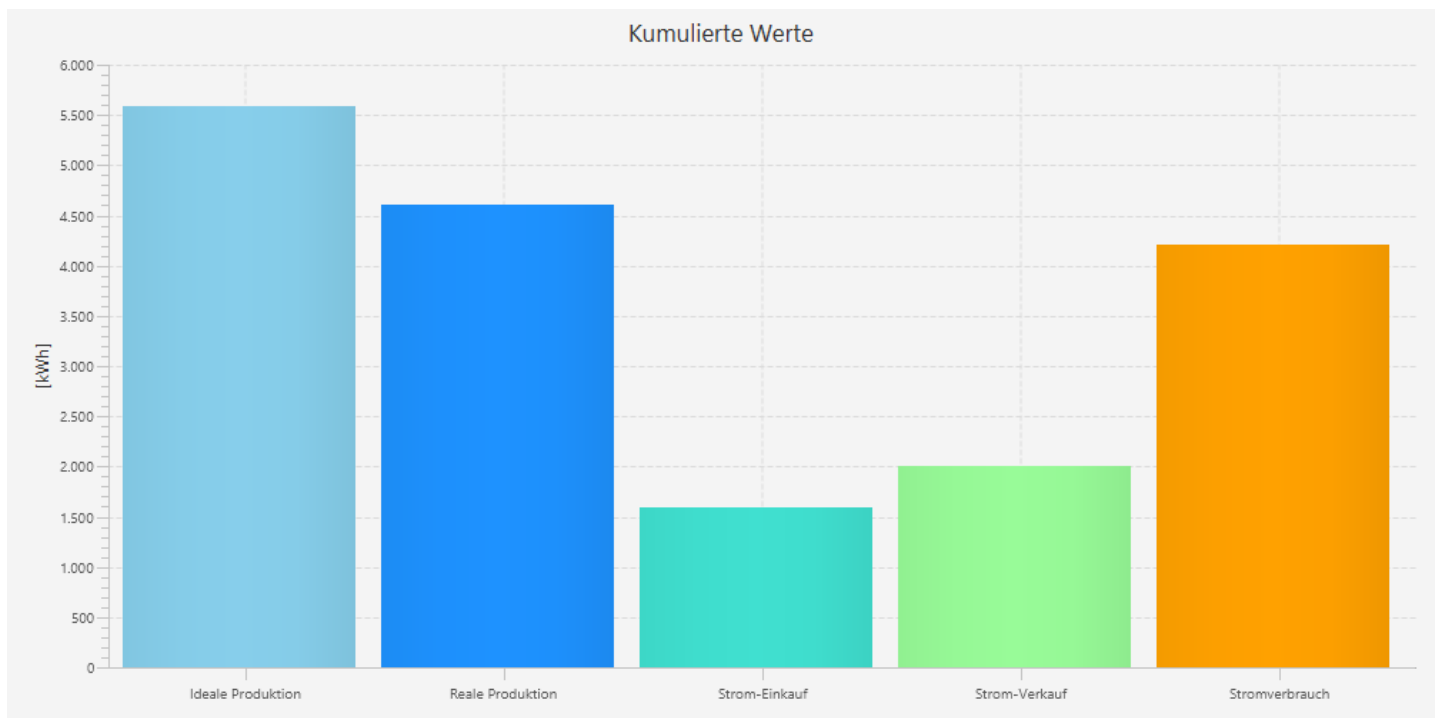
3.3. Übersicht Stromkosten für eine Woche im Januar



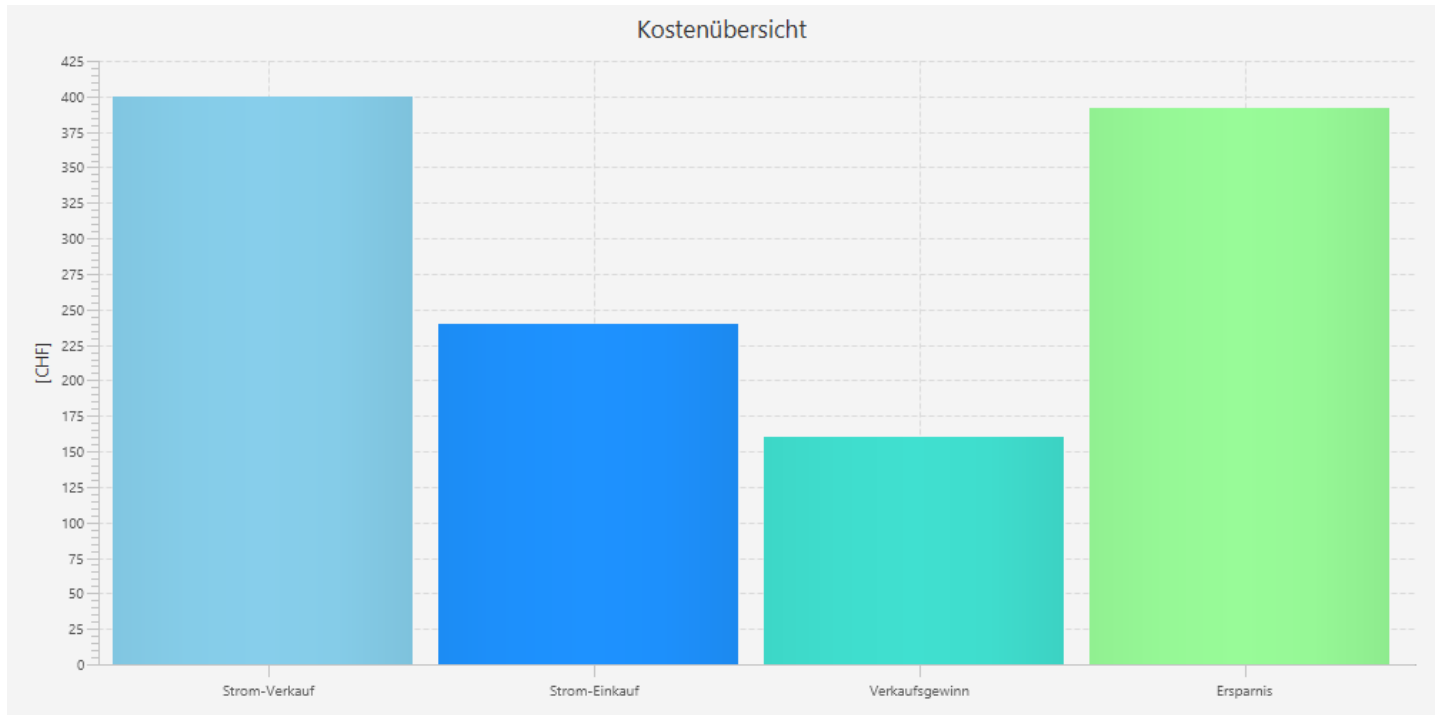
3.4. Jahresverlauf



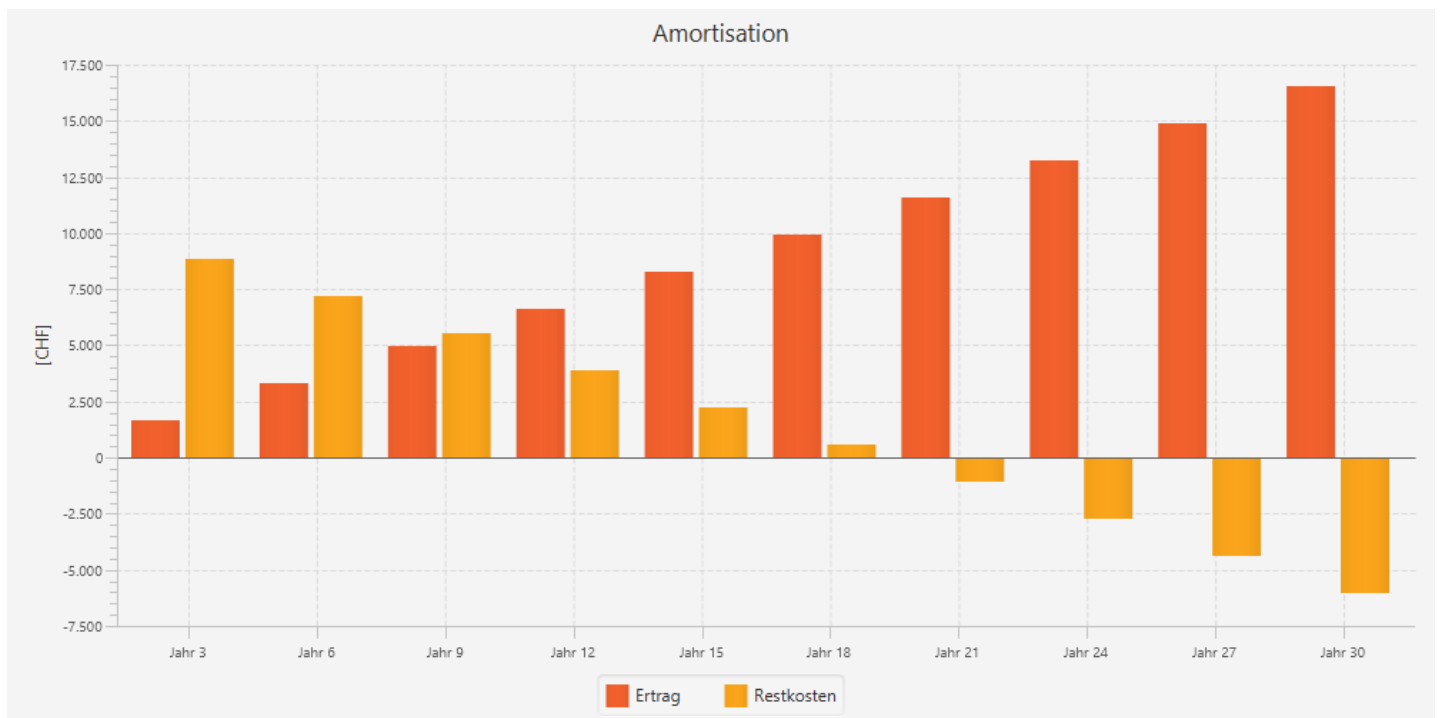
3.5. Jahresübersicht



3.6. Jahresübersicht der Stromkosten



3.7. Amortisation



CSV-Export

Aus der Applikation haben wir die Simulationsergebnisse als CSV exportiert. Die verwendeten Benutzerdaten sind in Kapitel 10.2 beschrieben.

Berechnungsergebnisse für den Zeitraum eines Jahres

Monat	Ideale Produktion [Wh]	Reale Produktion [Wh]	Produktionsüberschuss [Wh]	Batterie-Ladestand [Wh]	Eingekaufter Strom [Wh]	Verkaufter Strom [Wh]	Verbrauchter Strom [Wh]	Einstrahlung Horizontal [Wh/m2]	Einstrahlung Direkt [Wh/m2]	Einnahmen aus Verkauf	Stromeinkaufs-Kosten	Gewinn aus Verkauf	Eingesparte Strom-Kosten
Januar	197556.18	162885.47	-80400.67	0	80400.67	0	243286.14	58545.05	57362.4	0	12.09	-12.09	24.49
Februar	288451.98	237829.58	10239.32	0	46328.66	56567.98	227590.26	54767.95	83754.9	11.31	6.96	4.35	27.26
März	473274.52	390214.98	146928.84	0	26116.88	173045.72	243286.14	58545.05	137419.9	34.72	4.03	30.69	32.55
April	597727.2	492825.6	257387.4	6984	7388.4	263778	235438.2	56656.5	173556	52.8	1.2	51.6	34.2
Mai	705369.35	581576.74	338290.6	14939.83	0	337187.31	243286.14	58545.05	204810.8	67.58	0	67.58	36.58
Juni	720036.6	593670	358231.8	18257.4	0	357689.1	235438.2	56656.5	209070	71.4	0	71.4	35.4
Juli	738796.34	609137.6	365851.46	16718.61	0	366158.36	243286.14	58545.05	214516.9	73.16	0	73.16	36.58
August	652167.77	537712.67	294426.53	11565.17	0	295162.78	243286.14	58545.05	189363.5	58.9	0	58.9	36.58
September	495078.3	408191.7	172753.5	0	24155.7	198508.2	235438.2	56656.5	143751	39.6	3.6	36	31.8
Oktober	339755.35	280127.78	36841.64	0	50854.26	87695.9	243286.14	58545.05	98651.3	17.67	7.75	9.92	28.83
November	210142.8	173262.3	-62175.9	0	66246.9	4071	235438.2	56656.5	61017	0.9	9.9	-9	25.5
Dezember	160402.99	132252.51	-111033.63	0	111033.63	0	243286.14	58545.05	46574.4	0	16.74	-16.74	19.84

Berechnungsergebnisse für den Zeitraum einer Woche

Wochentag	Ideale Produktion [Wh]	Reale Produktion [Wh]	Produktionsüberschuss [Wh]	Batterie-Ladestand [Wh]	Eingekaufter Strom [Wh]	Verkaufter Strom [Wh]	Verbrauchter Strom [Wh]	Einstrahlung Horizontal [Wh/m²]	Einstrahlung Direkt [Wh/m²]	Einnahmen aus Verkauf	Stromeinkaufs-Kosten	Gewinn aus Verkauf	Eingesparte Strom-Kosten
Montag	6372.78	5254.37	-2593.57	0	2593.57	0	7847.94	1888.55	1850.4	0	0.3890355	-0.3890355	0.7881555
Dienstag	6372.78	5254.37	-2593.57	0	2593.57	0	7847.94	1888.55	1850.4	0	0.3890355	-0.3890355	0.7881555
Mittwoch	6372.78	5254.37	-2593.57	0	2593.57	0	7847.94	1888.55	1850.4	0	0.3890355	-0.3890355	0.7881555
Donnerstag	6372.78	5254.37	-2593.57	0	2593.57	0	7847.94	1888.55	1850.4	0	0.3890355	-0.3890355	0.7881555
Freitag	6372.78	5254.37	-2593.57	0	2593.57	0	7847.94	1888.55	1850.4	0	0.3890355	-0.3890355	0.7881555
Samstag	6372.78	5254.37	-2593.57	0	2593.57	0	7847.94	1888.55	1850.4	0	0.3890355	-0.3890355	0.7881555
Sonntag	6372.78	5254.37	-2593.57	0	2593.57	0	7847.94	1888.55	1850.4	0	0.3890355	-0.3890355	0.7881555

Jahrestotal

Ideale Produktion [Wh]	5578759.38
Reale Produktion [Wh]	4599686.93
Produktionsüberschuss [Wh]	1727340.89
Batterie-Ladestand [Wh]	68465.01
Eingekaufter Strom [Wh]	412525.1
Verkaufter Strom [Wh]	2139864.35
Verbrauchter Strom [Wh]	2872346.04
Einstrahlung Horizontal [Wh/m²]	691209.3
Einstrahlung Direkt [Wh/m²]	1619848.1
Einnahmen aus Verkauf	428.04
Stromeinkaufs-Kosten	62.27
Gewinn aus Verkauf	365.77
Eingesparte Strom-Kosten	369.61