



Pretty Secure Cloud

Zentrale Verschlüsselung für Cloud-Dienste



Lösungsarchitektur

Autoren: Guerotto Sandro, Kindle Tristan, Sevimli Ridvan, Sieber Lorenz, Waldburger Safiyya und
Walser Christoph Marjan Markus

Modul: Software Projekt 3, Frühlingssemester 2022

Dozierende: Liebhart Daniel, Schnatz Nina Isabelle

20. April 2021

Inhalt

1. Einleitung	1
2. Use-Case-Modell	1
2.1. Use Case 1: Verschlüsselung und Upload	2
2.1.1. Beschreibung	2
2.1.2. UI-Sketch	3
2.1.3. Sequenzdiagramm	3
2.2. Use-Case 2: Download und Entschlüsselung	5
2.2.1. Beschreibung	5
2.2.2. UI-Sketch	6
2.2.3. Sequenzdiagramm	6
2.3. Use-Case 3: Registrierung eines neuen Accounts	8
2.3.1. Beschreibung	8
2.3.2. UI-Sketch	8
3. Zusätzliche Anforderungen	9
3.1. Funktionalität (Functionality)	9
3.2. Gebrauchstauglichkeit (Usability)	9
3.3. Zuverlässigkeit (Reliability)	9
3.4. Performanz (Performance)	9
3.5. Unterstützbarkeit (Supportability)	9
4. Domänenmodell	10
5. Softwarearchitektur	11
5.1. Package-Diagramm	11
5.2. Class-Diagramm	12
6. Designkonzept	14
7. Implementation	15
8. Projektmanagement	16
8.1. Allgemeines	16
8.2. Projektplan	16
9. Organisatorisches	17
9.1. Teammitglieder	17
9.2. Aufgabenverteilung	17

10. Selbstständigkeitserklärung	18
11. Glossar	19
12. Quellenverzeichnis	21
13. Abbildungsverzeichnis.....	21
14. Tabellenverzeichnis.....	21

1. Einleitung

Die Pretty Secure Cloud (nachfolgend *PSC*) ist ein automatischer Ver- und Entschlüsselungsdienst für Dateien, welche in einem Cloud-Service eines Drittanbieters gespeichert werden sollen. Die Ver- und Entschlüsselungsfunktionalität kann unabhängig vom gewählten Speicherort verwendet werden.

Der vorliegende technische Bericht gibt Auskunft über den Prozess, Fortschritt und Resultate im Zusammenhang mit der Entwicklung der erwähnten Anwendung am Ende der dritten Entwicklungsiteration. Insbesondere befasst sich der technische Bericht mit den erarbeiteten Resultaten der Anforderungsanalyse und erörtert ebenfalls die geplante Architektur sowie Designentscheide. Des Weiteren wird der Fortschritt anhand der Iterationsreviews aufgezeichnet und etwaige Massnahmen hervorgehoben.

2. Use-Case-Modell

Im folgenden Abschnitt werden die identifizierten Use-Cases “Dateien verschlüsseln und hochladen” sowie “Dateien runterladen und entschlüsseln” ausführlich (d.h. *fully dressed*) erläutert, sowie die dazugehörigen UML-Diagramme präsentiert. Die beiden Use-Cases werden neben dem gemeinsamen Use-Case-Diagramm jeweils mittels einem System-Sequenzdiagramm, sowie einem UI-Sketch visualisiert. Der Use-Case “Registrierung eines neuen Accounts” wird als informeller Use-Case erläutert (d.h. *casual*).

Die Interaktion eines Benutzers und *PSC* wird im nachfolgenden Use-Case-Diagramm aufgezeigt (Abbildung 1 Use-Case-Diagramm). Daraus wird ersichtlich, dass die Hauptanwendungsfälle für die Benutzung “Verschlüsselung und Upload” und “Download und Entschlüsselung” sind. Weiter ist erkennbar, dass der/die Benutzer/-in das System initialisieren und gewisse Einstellung vornehmen muss (“Set up”).

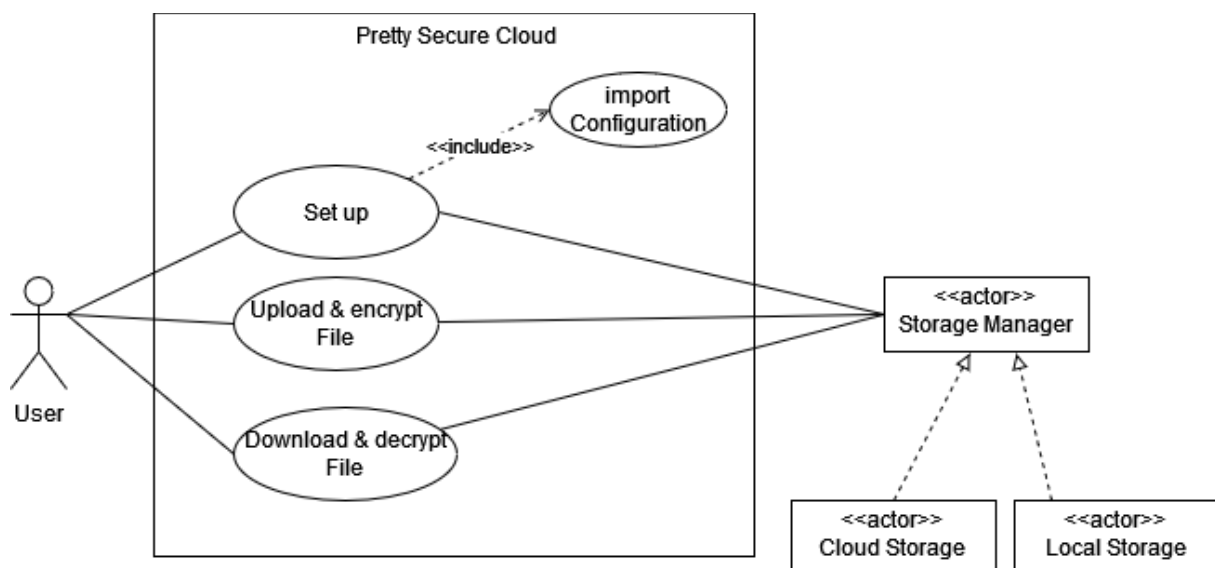


Abbildung 1 Use-Case-Diagramm

2.1. Use Case 1: Verschlüsselung und Upload

2.1.1. Beschreibung

Im Use-Case “Verschlüsselung und Upload” geht es um den Prozess, welcher durchlaufen werden muss, um ausgewählte Dateien mittels der PSC verschlüsseln und auf einen ausgewählten Cloud-Dienst hochzuladen.

Primärakteur/in	Cloudbenutzer/-in
Stakeholder / Interessen	Cloudbenutzer/-in will: <ul style="list-style-type: none"> • Daten verschlüsseln • Dateien schnell und problemlos auf Cloud laden
Vorbedingungen	<ul style="list-style-type: none"> • Benutzer/-in ist identifiziert und authentifiziert vom System • Gültige Logindaten bei Cloud-Dienst • Funktionierende Internetverbindung
Nachbedingungen	<ul style="list-style-type: none"> • Die Datei wurde verschlüsselt in einen Cloud-Dienst hochgeladen • Benutzer/-in ist bewusst, dass Dateien nur mit dem entsprechenden Schlüssel wieder entschlüsselt werden können
Hauptablauf	<ul style="list-style-type: none"> • Benutzer/-in wählt Dateien zum Verschlüsseln und Hochladen aus [Alt1: Dateien mit selben Namen befinden sich in dem Cloud-Verzeichnis] • Das System erhält die nötigen Angaben, um die Dateien zu verschlüsseln und auf die Cloud zu laden. • System prüft Cloud-Anbindung [Alt2: Zugriff auf Cloud nicht möglich] • System benachrichtigt Benutzer/-in über den Status des Uploads • System bietet Möglichkeit weitere Dateien hochzuladen [Alt 3: Benutzer/-in möchte weitere Dateien hochladen] oder Use Case zu verlassen [Use Case beendet]
Alternative Abläufe	<p>Alt1:</p> <ul style="list-style-type: none"> • System informiert Benutzer/-in, dass Dateien mit selben Namen sich in Cloud-Verzeichnis befinden • System bietet Möglichkeit, Dateien, auf der Cloud zu überschreiben, eindeutige Name zu vergeben oder die Aktion abubrechen. [Use Case beendet] • Hauptablauf Punkt 2 <p>Alt 2:</p> <p>Benutzer/-in wird informiert über fehlerhafte Cloud-Anbindung [Use Case beendet]</p> <p>Alt 3:</p> <p>Hauptablauf, Punkt 1</p>
Spezielle Anforderungen	<ul style="list-style-type: none"> • Upload Möglichkeit via Cloud API • Sichere Aufbewahrungsmethode für Schlüssel
Häufigkeit des Auftretens	Abhängig von Intensität der Cloud-Nutzung

Tabelle 1 Use-Case Verschlüsselung und Upload

2.1.2. UI-Sketch

Im nachfolgenden UI-Sketch wird der behandelte Use-Case anhand von beispielhaften Skizzen der möglichen Benutzeroberfläche visualisiert (Abbildung 2 UI-Sketch Verschlüsselung und Upload). Der/die Benutzer/-in gelangt zum Hauptfenster (links), welches eine Übersicht über die hochzuladenden Dateien inkl. Statusanzeige gibt. Dort ist ersichtlich, dass die Selektion der hochzuladenden Dateien über ein Drag and Drop Fenster vorgenommen werden kann.

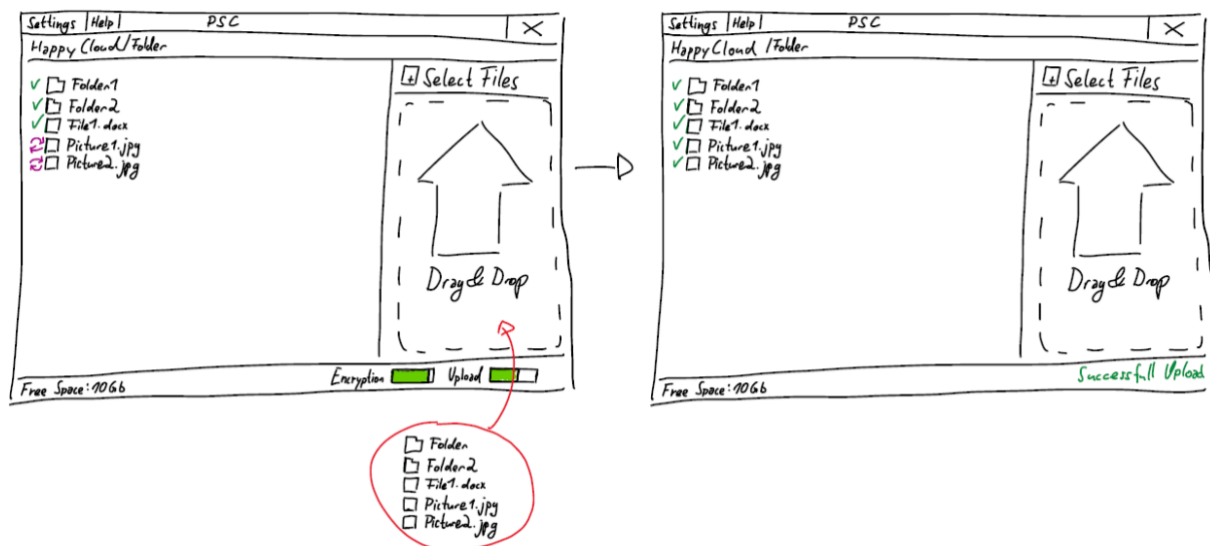


Abbildung 2 UI-Sketch Verschlüsselung und Upload

2.1.3. Sequenzdiagramm

Das nachfolgende Sequenzdiagramm veranschaulicht den Use-Case "Verschlüsselung und Upload" mit Blick auf das Verhalten der involvierten Systeme untereinander (Abbildung 3 Sequenzdiagramm Verschlüsselung und Upload). Im Zentrum stehen insbesondere die sogenannten Lebenslinien eines Systems (vertikal), welche die Aktivität des jeweiligen Systems visualisieren, d.h. wann sie aktiv werden (längliche Rechtecke) respektive inaktiv bleiben, sofern sie nicht interagieren (gestrichelte Linien).

Das Diagramm wird grundsätzlich von oben nach unten gelesen, beginnend beim "User" (oben links), welcher eine oder mehrere Dateien für die Verschlüsselung und den Upload im "Client" selektiert. Der "Client" gibt die erhaltene Datei an den "StorageManager" weiter, welcher sich nun zentral um den Prozess kümmert. Dieser übergibt die Datei zunächst an den "Encrypter", welcher versucht die Datei zu verschlüsseln. Das Resultat wird dann an den "StorageManager" zurückgegeben, welcher die Verschlüsselung dem "Client" bestätigt oder ablehnt. Im letzteren Fall kommt der Vorgang bereits zu einem Ende, da keine verschlüsselte Datei für den Upload zur Verfügung steht. War die Verschlüsselung erfolgreich, so leitet der "StorageManager" die verschlüsselte Datei an den "CloudService" weiter (Upload). Der "StorageManager" wartet dann auf eine Antwort vom "CloudService", ob der Upload erfolgreich war oder nicht und übergibt die erhaltene Information zurück an den "Client", welcher den Status für die entsprechende Datei aktualisiert. Hervorzuheben ist, dass der "User" nach dem Drag and Drop der Datei keine weiteren Interaktionen mit den involvierten

Systemen vornehmen muss. Grundsätzlich kann der "User" aber jederzeit den Status der Verschlüsselung und des Uploads im "Client" nachvollziehen.

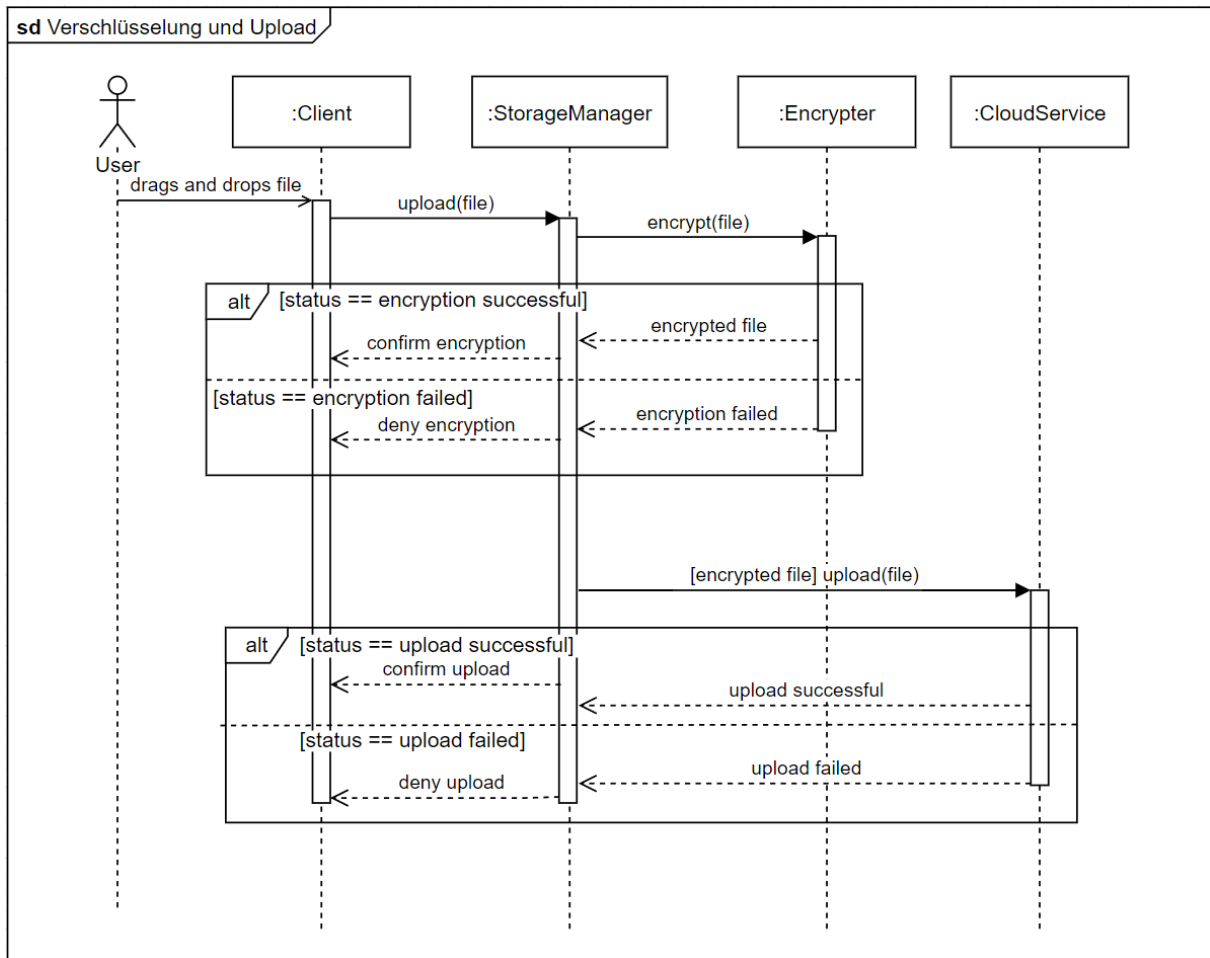


Abbildung 3 Sequenzdiagramm Verschlüsselung und Upload

2.2. Use-Case 2: Download und Entschlüsselung

2.2.1. Beschreibung

Im Use-Case “Download und Entschlüsselung” geht es um den Prozess, welcher ein/e Benutzer/-in durchlaufen muss, wenn er/sie bereits verschlüsselte und hochgeladene Dateien (siehe Ziffer 2) mittels PSC vom jeweiligen Speicherort herunterladen und entschlüsseln möchte.

Primärakteur/in	Cloudbenutzer/-in
Stakeholder / Interessen	Cloudbenutzer/-in will: <ul style="list-style-type: none"> • Zugriff auf seine/ihre Dateien • Dateien entschlüsseln.
Vorbedingungen	<ul style="list-style-type: none"> • Benutzer/-in ist identifiziert und authentifiziert vom System • Gültige Logindaten bei Cloud-Diensten • Gültiger Schlüssel für Entschlüsselung
Nachbedingungen	Entschlüsselte Dateien sind an gewünschtem Ort gesichert
Hauptablauf	<ul style="list-style-type: none"> • System prüft Cloudanbindung [Alt1: Zugriff auf Cloud nicht möglich] • System zeigt verschlüsselte Dateien/Ordner, die sich auf der Cloud befinden • Benutzer/-in wählt Dateien für den Download und die Entschlüsselung • Das System erhält die nötigen Angaben, um die Dateien zu entschlüsseln und lokal zu speichern. • System benachrichtigt Benutzer/-in über erfolgreichen Download und Entschlüsselung • System bietet Möglichkeit weitere Dateien runterzuladen [Alt 3: User möchte weitere Dateien hochladen] oder Use Case zu verlassen [Use Case beendet]
Alternative Abläufe	<p>Alt 1: Benutzer/-in wird informiert über fehlerhafte Cloudanbindung [Use Case beendet]</p> <p>Alt 2:</p> <ul style="list-style-type: none"> • System informiert Benutzer/-in, dass Dateien mit selben Namen sich in dem lokalen Ordner befinden • System bietet Möglichkeit, lokale Dateien zu überschreiben, eindeutige Name zu vergeben oder die Aktion abzubrechen. [Use Case beendet] • Hauptablauf Punkt 2 <p>Alt 3: Hauptablauf, Punkt 1</p>
Spezielle Anforderungen	<ul style="list-style-type: none"> • Downloadmöglichkeit via API der Clouddienste • Sichere Aufbewahrungsmethoden für Schlüssel
Häufigkeit des Auftretens	Abhängig von Intensität der Cloudnutzung

Tabelle 2 Use-Case Download und Entschlüsselung

2.2.2. UI-Sketch

In dem nachfolgenden UI-Sketch wird der behandelte Use-Case anhand von beispielhaften Skizzen des möglichen Benutzerinterfaces visualisiert (Abbildung 4 UI-Sketch Download und Entschlüsselung). Beginnend links, wählt der/die Benutzer/-in die Dateien aus, welche vom aktuellen Speicherort heruntergeladen und entschlüsselt werden sollen (dargestellt mit einem grünen Rechteck). Im nächsten Fenster (rechts) erhält der/die Benutzer/-in eine Übersicht über die herunterzuladenden Dateien inkl. Statusanzeige. Während dem Prozess, wird der Fortschritt für den Download und die Entschlüsselung unten rechts im Fenster mittels Statusanzeigen visualisiert.

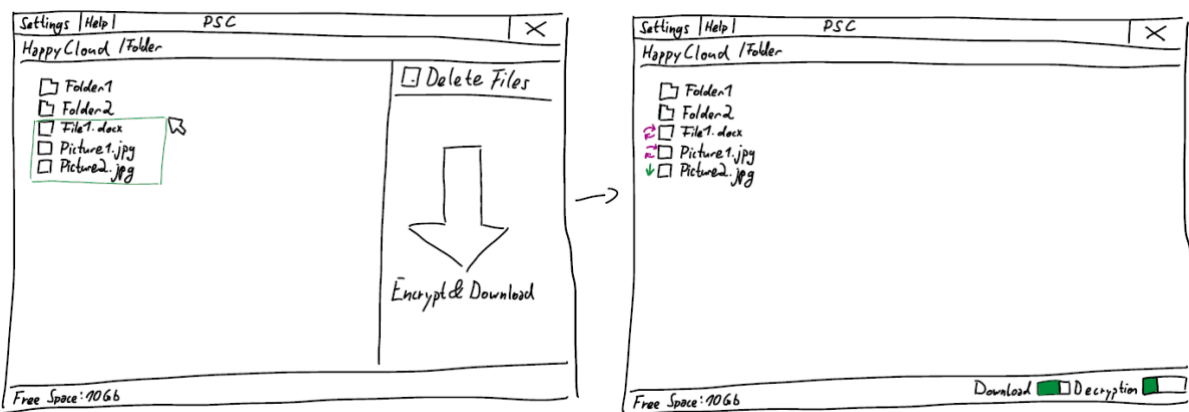


Abbildung 4 UI-Sketch Download und Entschlüsselung

2.2.3. Sequenzdiagramm

Das nachfolgende Sequenzdiagramm veranschaulicht den Use-Case "Download und Entschlüsselung" mit Blick auf das Verhalten der involvierten Systeme untereinander (Abbildung 5 Sequenzdiagramm: Download und Entschlüsselung). Im Zentrum stehen insbesondere die sogenannten Lebenslinien eines Systems (vertikal), welche die Aktivität des jeweiligen Systems visualisieren, d.h. wann sie aktiv werden (längliche Rechtecke) respektive inaktiv bleiben, sofern sie nicht interagieren (gestrichelte Linien). Das Diagramm wird grundsätzlich von oben nach unten gelesen, beginnend beim "User" (oben links), welcher eine oder mehrere Dateien für den Download und die Entschlüsselung im "Client" selektiert. Der "Client" gibt die erhaltene Dateiauswahl an den "StorageManager" weiter, welcher sich nun zentral um den Prozess kümmert. Dieser übergibt die Dateiauswahl zunächst an den "CloudService" und beantragt den Download der angeforderten Dateien. Das Resultat des Downloads wird anschliessend an den "StorageManager" zurückgegeben, welcher den Download dem "Client" bestätigt oder ablehnt. Im letzteren Fall kommt der Vorgang bereits zu einem Ende, da keine verschlüsselten Dateien für die Entschlüsselung zur Verfügung stehen. War das Herunterladen erfolgreich, so leitet der "StorageManager" die verschlüsselte Datei an den "Decrypter" weiter. Der "StorageManager" wartet dann auf eine Antwort vom "Decrypter", ob die Entschlüsselung erfolgreich war oder nicht und übergibt die erhaltene Information zurück an den "Client", welcher den Status für die entsprechende Datei aktualisiert. Hervorzuheben ist, dass der "User" nach dem Selektieren der Dateien keine weiteren Interaktionen mit den involvierten Systemen vornehmen muss. Grundsätzlich kann der "User" aber jederzeit den Status des Downloads und der Entschlüsselung im "Client" nachvollziehen.

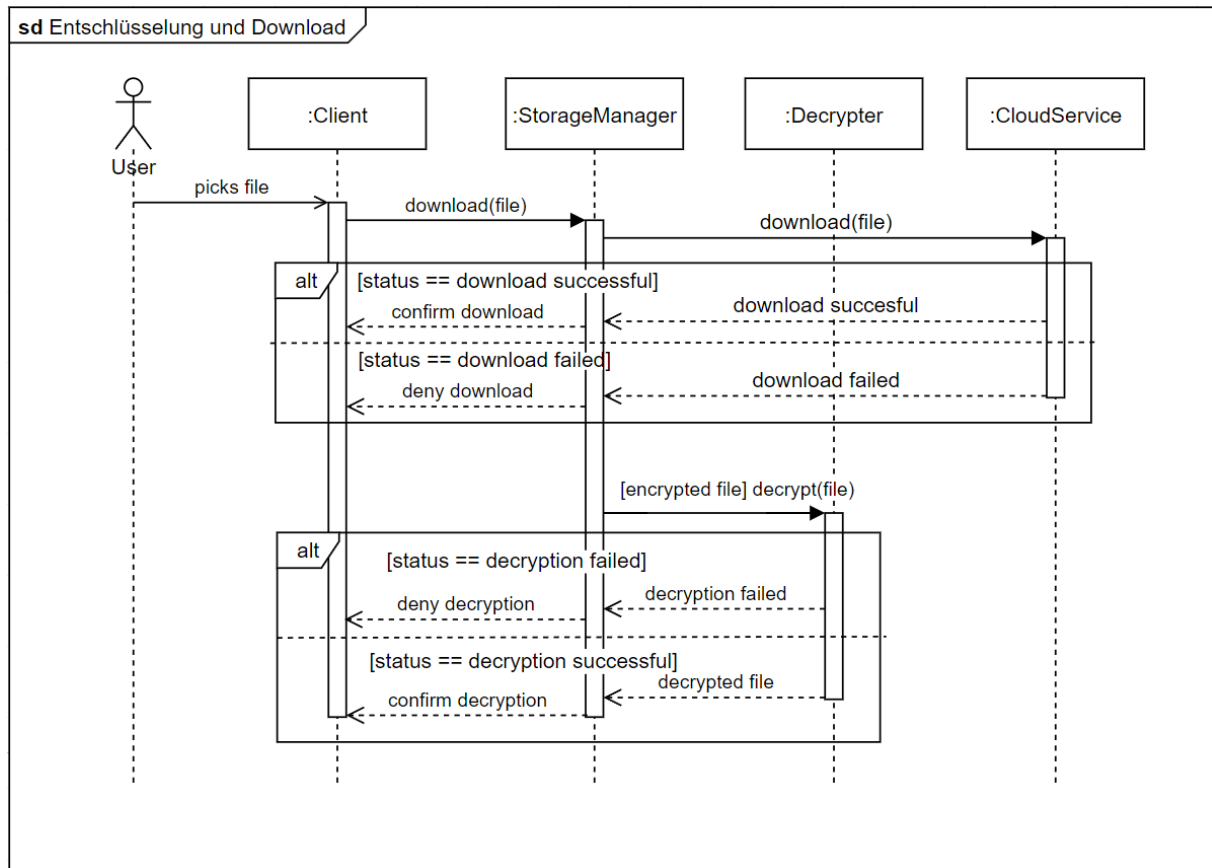


Abbildung 5 Sequenzdiagramm: Download und Entschlüsselung

2.3. Use-Case 3: Registrierung eines neuen Accounts

2.3.1. Beschreibung

Erfolgsszenario	<ul style="list-style-type: none"> • Zur Registrierung bei dem Cloudanbieter werden die Login-Daten angegeben. • Nach dem Absenden und Überprüfen der Login-Daten wird dem/der Benutzer/-in zurückgemeldet, ob die Registrierung erfolgreich war. • Falls Registrierung nicht erfolgreich war, wird der/die Benutzer/-in aufgefordert, Login-Daten nochmals einzugeben.
-----------------	--

Tabelle 3 Use-Case Registrierung eines neuen Accounts

2.3.2. UI-Sketch

Auf der nachfolgenden Abbildung ist die Auswahl des Cloud-Anbieters ersichtlich (Abbildung 6 UI-Sketch Registrierung eines neuen Accounts). Nach der Auswahl folgt die Eingabe der Benutzerdaten, welche auf der zweiten Abbildung ersichtlich ist.

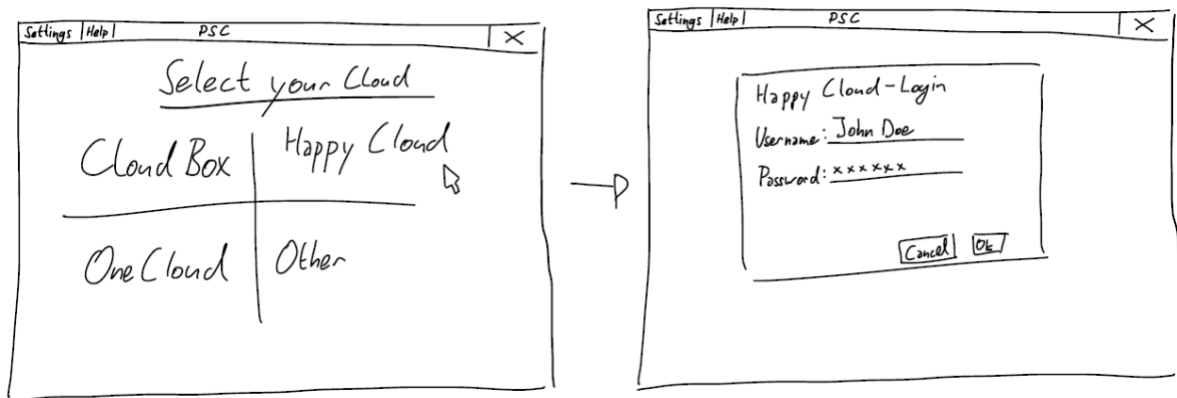


Abbildung 6 UI-Sketch Registrierung eines neuen Accounts

3. Zusätzliche Anforderungen

In diesem Kapitel werden die zusätzlichen Anforderungen nach dem vereinfachten FURPS-Modell erläutert.

3.1. Funktionalität (Functionality)

Die Hauptfunktionalitäten der Anwendung ist das Ver- und Entschlüsseln von Dateien sowie das Hoch- und Herunterladen auf bzw. von verbundenen Speicherdiensten. Als zusätzliche Funktionen ist die Möglichkeit von **Drag and Drop** im Zusammenhang mit Dateibewegungen über die Benutzeroberfläche. Ausserdem wird der **Status** der erwähnten Hauptfunktionalität pro Datei in der Benutzeroberfläche visualisiert.

Um die Wiederverwendbarkeit der Dateien, z.B. wenn der spezifische Schlüssel für den verwendeten Verschlüsselungsalgorithmus verloren geht, zu gewährleisten, werden **nur gebräuchliche Verschlüsselungsalgorithmen** eingesetzt. Dadurch kann die Entschlüsselung der Dateien grundsätzlich auch von Drittanwendungen vorgenommen werden. Durch die Verwendung von bekannten und modernen Verschlüsselungsalgorithmen wird eine grösstmögliche **Datensicherheit** angestrebt.

3.2. Gebrauchstauglichkeit (Usability)

Die Benutzeroberfläche ist nach dem "Keep it simple and stupid (KISS)"-Prinzip aufgebaut. Somit wird gewährleistet, dass die Anwenderinnen und Anwender **keine Bedienungsanleitung** oder dergleichen benötigen, um die PSC-Applikation zu bedienen. Als Regel wird festgelegt, dass der/die Benutzer/-in innert "2 Minuten" die Anwendung installieren, einstellen und verwenden kann. Dies wird insbesondere durch einen **schlanken Aufbau der Benutzeroberflächen** erreicht.

3.3. Zuverlässigkeit (Reliability)

Durch die Verwendung von **ausschliesslich modernen Verschlüsselungsalgorithmen** wird ein zuverlässiger Verschlüsselungs- und Entschlüsselungsprozess für alle gebräuchlichen Dateitypen gewährleistet. Sollte zu einem gegebenen Zeitpunkt keine aktive Internetverbindung vorliegen, können die verschlüsselten **Dateien auch lokal gespeichert** werden. Bei Bedarf können diese zu einem späteren Zeitpunkt auf einen angebundenen Speicherdienst hochgeladen werden.

3.4. Performanz (Performance)

Um unerwünschte Verzögerung oder gar Abstürze während der Benutzung der Anwendung zu vermeiden, wird der Prozess der Ver- und Entschlüsselung sowie des Hoch- und Herunterladens in **einzelne Threads pro Datei** zerlegt. Wenn ein Prozess fehlschlägt, wird dies in der Benutzeroberfläche angezeigt und der/die Benutzer/-in kann diesen erneut initiieren.

3.5. Unterstützbarkeit (Supportability)

Im gesamten Softwareentwicklungsprozess sollten fortlaufend Tests für die Klassen geschrieben werden, so dass bei Änderungen einer Klasse, diese direkt für die geforderte Funktionalität getestet werden kann. Bei jedem Antrag auf Programmänderung werden sämtliche **Tests automatisch von**

GitHub durchgeführt und gegebenenfalls abgelehnt, sollte einer der Tests fehlschlagen (siehe Ziffer 15).

4. Domänenmodell

Das Domänenmodell (Abbildung 7 Domänenmodell Pretty Secure Cloud) repräsentiert die Abhängigkeiten in der Applikation. Eine zentrale Rolle in der Domäne nimmt der/die Benutzer/-in ("User") und die verknüpften Speicherdienste ("StorageManager") ein. Vereinfacht formuliert, steuert der "User" über den "StorageManager" das Hochladen (inkl. Verschlüsselung) sowie das Herunterladen (inkl. Entschlüsselung) der ausgewählten Datei. Der "StorageManager" verwaltet die möglichen Speicherorte ("CloudStorage" und "LocalStorage") und koordiniert in dieser Position das Hoch- und Herunterladen auf/von den erwähnten Speicherorten. Für jeden "User" können ein (symmetrische Verschlüsselung) oder zwei (asymmetrische Verschlüsselung) Schlüssel ("Key") angelegt werden, welche sodann für alle Verschlüsselungs- und Entschlüsselungsvorgänge verwendet werden ("Encryption" und "File"). Aus dem Domänenmodell ist weiter ersichtlich, dass für jede Datei der Verschlüsselungszustand gemerkt wird ("EncryptionState").

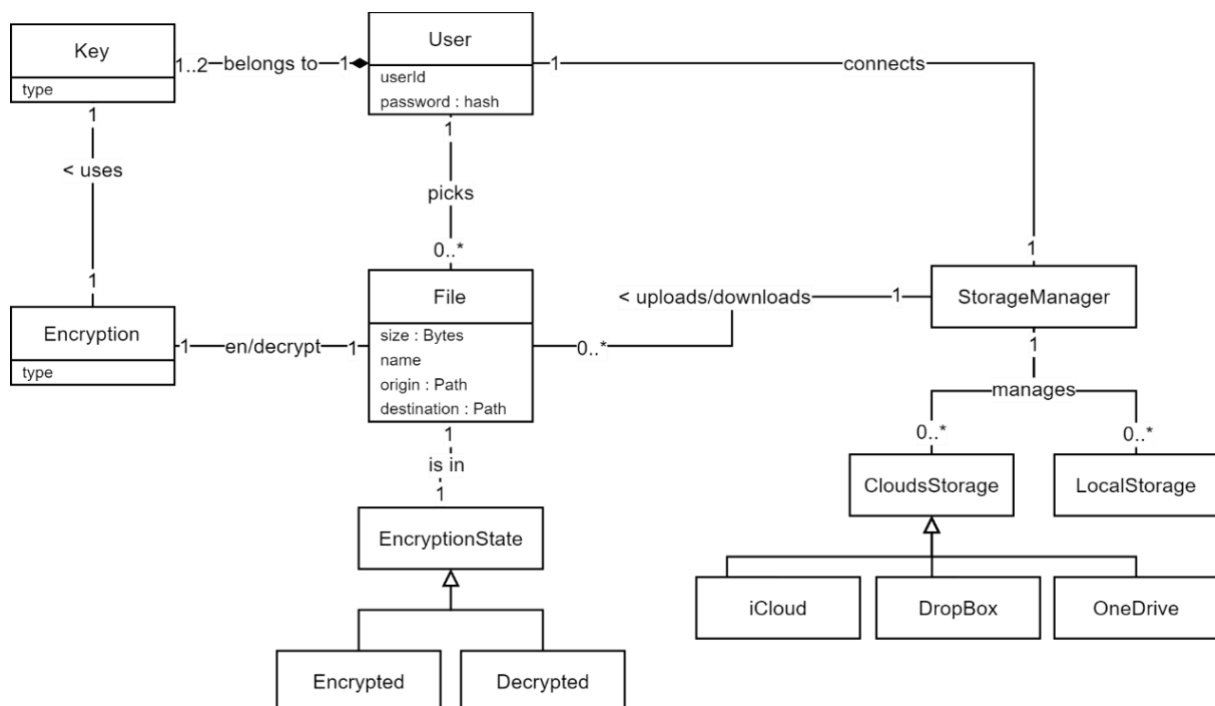


Abbildung 7 Domänenmodell Pretty Secure Cloud

5. Softwarearchitektur

5.1. Package-Diagramm

Für die Applikation wird eine *three-tier-architecture* umgesetzt. Das heisst, die Applikation wird in drei Schichten aufgeteilt: Präsentation ("presentation"), Logik ("domain") und Datenhaltung ("datasources"). Dabei darf jeweils nur die obere Schicht auf die untere zugreifen, wobei das Überspringen einer Schicht verboten ist. Dieses Architekturprinzip zeichnet sich dadurch aus, dass eine beliebige Schicht ausgetauscht werden kann und dies einen geringen oder bestenfalls keinen Einfluss auf die übrigen Schichten hat.

Das nachfolgende Package-Diagramm (Abbildung 8 Package-Diagramm) visualisiert, die im obigen Absatz beschriebene Programmstruktur mit sogenannten *packages*, sowie deren Interaktionen bzw. Abhängigkeiten untereinander. Das Package-Diagramm gibt insbesondere Einsicht in die verschiedenen Schichten der Programmarchitektur und Auskunft über die verwendeten Frameworks. Die Anwendung als Ganzes wird mit dem *package* "ch.psc" dargestellt.

Ausgehend vom *package* "presentation" ist zu erkennen, dass das Java-Framework JavaFX für die Präsentationsschicht verwendet wird. JavaFX eignet sich insbesondere für die Umsetzung des angestrebten "Model-View-Controller (MVC)"-Patterns, welches nachfolgend erläutert wird. Der sogenannte Controller ist für die Kommunikation zwischen den Benutzeroberflächen (View) und der Domäne (Model) zuständig und wird als zentraler Programmteil im Paket "controller" innerhalb des Pakets "presentation" abgelegt. Die Benutzeroberflächen (sogenannte Views) werden als FXML-Dateien umgesetzt. Die Verwendung von FXML-Dateien für die Entwicklung der Benutzeroberflächen erlaubt den Gebrauch des grafischen Tools SceneBuilder von Gluon, welcher wiederum die Entwicklung erheblich unterstützen kann.

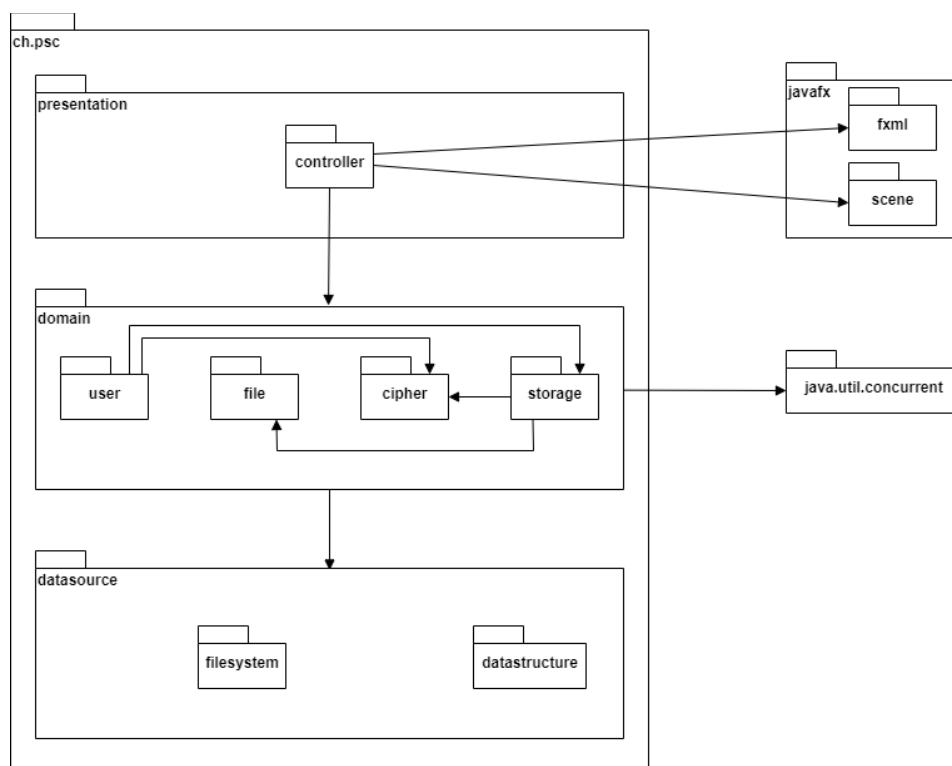


Abbildung 8 Package-Diagramm

Was das Model aus dem “Model-View-Controller (MVC)“-Pattern betrifft, so weicht der obige Architekturentwurf von dem vorherrschenden Standard ab. In der Regel ist die Aufgabe des Models die Datenverwaltung der von der Benutzeroberfläche dargestellten Daten. Zudem wird das Model üblicherweise auf der Präsentationsschicht erstellt und ist folglich losgekoppelt von der Logikschicht “domain” der Anwendung. In diesem Fall werden aber für das Model einzelne Klassen direkt aus der “domain” verwendet. Dadurch lässt sich der Zusatzaufwand für beinahe identische Klassen auf der Präsentationsschicht und für den Datentransport verantwortlicher Objekte (DTO) einsparen. Das Projekt ist ausreichend klein, um diese erhöhte Koppelung der Schichten verantworten zu können.

Die Logikschicht “domain” umfasst die eigentliche Programmlogik (Verwaltung von Dateien, Ver- und Entschlüsselung, Benutzerdaten) und wird im nächsten Abschnitt genauer beleuchtet.

In der untersten Schicht (“datasource”) befinden sich Datenstrukturen “datastructure”, die in der Logikschicht verwendet werden, sowie Klassen, die für den Zugriff auf das Dateisystem verwendet werden (“filesystem”). Falls in Zukunft eine Datenbankbindung implementiert wird, wird diese auch dem *packages* “datasource” angehängt werden.

5.2. Class-Diagramm

Das Class-Diagramm visualisiert die aktuelle Programmarchitektur mit Blick auf die verwendeten Klassen sowie deren Eigenschaften, Methoden und Interaktionen untereinander (Abbildung 9 Class-Diagramm).

In der Logikschicht “domain” spielen sich mehrere Prozesse ab, welche über eine längere Zeit andauern können. Java bietet eine Bibliothek “java.util.concurrent”, die geeignete Methoden beinhaltet, um diese Prozesse asynchron auszuführen und zu einem späteren Zeitpunkt wieder zusammenzuführen. Voraussichtlich wird “java.util.concurrent” an mehreren Stellen im Programm verwendet werden, wie bspw. von den Interfaces “Cipher” und “FileStorage”.

Pro “User” existiert genau ein “StorageManager”, welcher für den “User” die Dateien verwaltet. Somit kann die Dateiverwaltung von der Klasse “User” wegabstrahiert werden, vorausgesetzt eine starke Beziehung, sogenannte Komposition, zwischen den beiden Klassen existiert. Der “StorageManager” weiss, wo sich die unterschiedlichen Dateien “File” befinden, und gibt die Ver- oder Entschlüsselung der Dateien in Auftrag (durch Verwendung von verschiedenen “Cipher” Implementationen).

Um unterschiedliche Verschlüsselungsverfahren anzubieten, wird das Interface “Cipher” definiert. Der Schlüssel “Key” selbst beinhaltet die Verschlüsselungsmethode, sodass die “CipherFactory” mithilfe des Schlüssels automatisch die dazugehörige Implementation des “Ciphers” erzeugen kann [1].

Damit wird gewährleistet, dass ziemlich einfach neue Verschlüsselungsmethoden angeboten werden können, ohne Änderungen am restlichen Code vornehmen zu müssen.

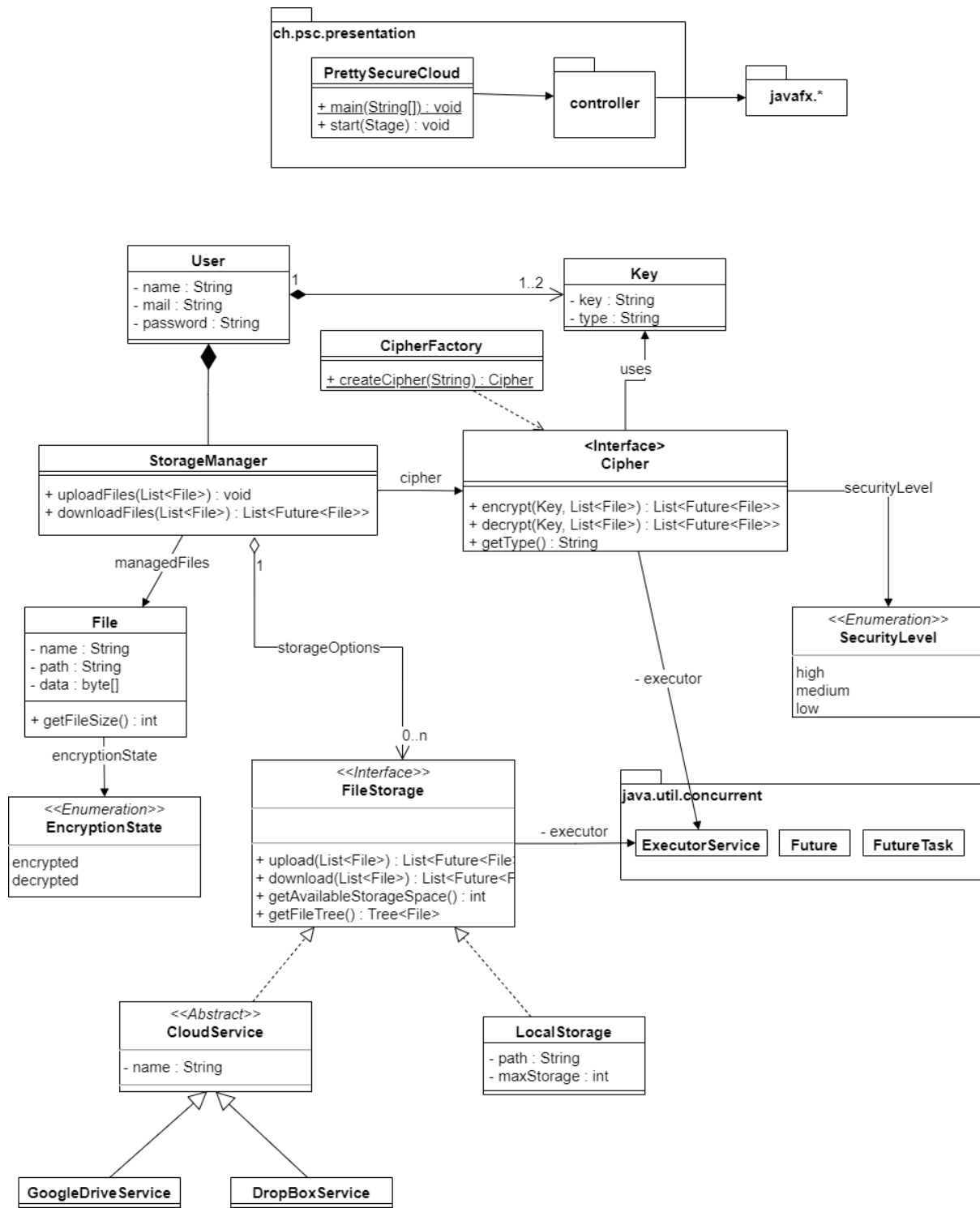


Abbildung 9 Class-Diagramm

6. Designkonzept


Die einheitliche Präsentation sämtlicher Programmelemente, welche für den/die Benutzer/-in sichtbar sind, werden basierend auf der folgenden Farbpalette entworfen (Tabelle 4 Farbpalette). Damit das Programm ein modernes *Look-and-Feel* hat, wird die moderne Schriftart Roboto verwendet.


HEX-Code	Verwendung
E1ECF7	Akzent III
AECBEB	Akzent II
83B0E1	Akzent I
71A5DE	Hauptakzent (Hover: Knopf)
4A4A4A	Hauptschrift
717171	Sekundärschrift (Vorschau/Icon)
EE4035	Status: Fehler
F37736	Status: Warnung II
FDF498	Status: Warnung I
7BC043	Status: OK


Tabelle 4 Farbpalette


Zudem werden wichtige Knöpfe und Eingabefelder mit einem einheitlichen Set von Symbolen unterstützt (Abbildung 10 Beispiel: Registrierungsformular).

Setup your account











 Back
 ☒
☐
☐
 Next 

Abbildung 10 Beispiel: Registrierungsformular

7. Implementation

Bei unserer Teststrategie haben wir uns für eine “Continuous Integration” entschieden. Dies bedeutet insbesondere, dass nur getestete und funktionsfähige Programmteile von der Versionskontrolle akzeptiert werden. Dies wird dadurch erreicht, dass die definierten Tests automatisch und für jede vorgeschlagene Änderung am Programm durchgeführt werden.

In unserem Fall wird diese angestrebte Testroutine mit den sogenannten *Actions* in GitHub umgesetzt. Wir haben uns für diese Teststrategie entschieden, weil es bei jeder Pull-Anfrage auf GitHub (d.h. Anfrage für Programmänderung) die implementierten Tests automatisch ausgeführt werden und die Anfrage nur genehmigt wird, wenn alle Tests erfolgreich bestanden wurden. Durch diesen automatisierten Prozess für das kontinuierliche Testen und das Zusammenführen der gemachten Änderungen mit dem aktuellen Programmteilen, wird eine höhere Qualität des Projektes gewährleistet. Entsprechend kann effizienter und kostengünstiger entwickelt werden, da das Testen und Integrieren automatisiert durchgeführt wird.

8. Projektmanagement

8.1. Allgemeines

Für die Projektleitung ist Tristan Kindle und Ridvan Sevimli (Stellvertretung) zuständig. Zum Aufgabenbereich der Projektleitung gehört die Dokumentation der Projektplanung sowie die zeitliche und fachliche Überwachung des Projektfortschritts. Weiter nimmt die Projektleitung sämtliche Aufgaben im Zusammenhang mit organisatorischem Charakter wahr (z.B. Ansetzen von Besprechungen).

Iterationsreviews: Der unten dargestellte Gesamtprojektplan (inkl. Meilensteinen, Tabelle 5) soll in zweiwöchentlichen Iterationen überprüft und umgesetzt werden. Die Protokolle der Iterationsreviews werden nach jedem Zyklus mit der Grobplanung abgeglichen und allfällige Abweichungen notiert und mit entsprechenden Massnahmen angegangen.

Zeitliches Budget: Für die Entwicklung einer ersten Version wird ein Aufwand von rund 500 Personenstunden erwartet. Die gemachten Angaben zum zeitlichen Budget werden regelmässig und mündlich innerhalb des Teams besprochen, um unvorhergesehene Aufwände zu erkennen.

Aufgabenverwaltung: Die detaillierten Aufgaben betreffend die Implementation werden via GitHub koordiniert und verwaltet: <https://github.com/SandroGuerotto/PrettySecureCloud/projects/1>
Diese werden wöchentlich, d.h. nicht nur in jedem Iterationsreview, im Team besprochen und vorangetrieben. Dadurch sollen Probleme frühzeitig erkannt und behoben werden.

8.2. Projektplan

SW	Meilenstein	Iteration	Aufwand in Stunden		Aufträge	Bemerkung nach Iterationsreview
			Soll	Ist		
1		-	40	40	Projektidee finden, Architektur skizzieren, Projektskizze ausarbeiten, Mockup, Präsentation vorbereiten	Alle Aufträge gemäss Grobplanung und im zeitlichen Budget erfüllt.
3	Projektskizze & Präsentation	1	20	20	Anforderungen, Ziele und Vision formulieren und präsentieren	Alle Aufträge gemäss Grobplanung und im zeitlichen Budget erfüllt.
5		2	60	60	UC detailliert ausformuliert, zusätzliche Anforderungen definiert, UI-Prototyp, Entwurf Domänenmodell	Organisatorische Umlage der Aufträge, d.h. Abweichung vom Grobplanung nicht aber vom zeitlichen Budget. Implementation für UI-Prototyp auf SW 7 verschoben. Dafür erster Entwurf für 'Technischer Bericht I' erstellt.
7		3	80	80	Domänenmodell fertig, Architektur stabil und als PoC verifiziert, Verfassen technischer Bericht	Alle Aufträge gemäss Grobplanung und im zeitlichen Budget erfüllt. UI-Prototyp (aus vorheriger Iteration) erstellt. Architektur wurde verifiziert, keine <i>showstopper</i> erkannt. Technischer Bericht I bereit für finale Kontrolle.
5	Lösungsarchitektur, Technischer Bericht 1	4	20		Architektur verifiziert, dokumentiert und präsentiert.	
6		5	100		UI Prototyp implementiert, UC realisiert und getestet, Mögliche Showstopper erkennen, Aktuelle <i>issues</i> gemäss GitHub lösen	
9		6	80		Technischer Bericht II erstellen, Aktuelle <i>issues</i> gemäss GitHub lösen	
11		7	80		Tests formuliert und erfüllt, Aktuelle <i>issues</i> gemäss GitHub lösen Technischer Bericht II fertigstellen,	
13	Beta Release & Technischer Bericht 2	8	20		Betaversion fertig implementiert, inkl. Systemtests und Dokumentation	

Tabelle 5 Projektplan

9. Organisatorisches

9.1. Teammitglieder

Guerotto Sandro, Kindle Tristan, Sevimli Ridvan, Sieber Lorenz, Waldburger Safiyya und Walser Christoph Marjan Markus.

9.2. Aufgabenverteilung

Die Aufgabenverteilung für das Projekt wurde nicht statisch festgesetzt, d.h. die Aufgaben werden bei Bedarf umverteilt und grundsätzlich werden die Teammitglieder für alle Bereiche eingesetzt. Dadurch wird eine ausgewogene Verteilung der Arbeitslast realisiert. Die Implementation im Besonderen wird in Zweierteams realisiert. Die verschiedenen Aufgabenbereiche wurden prinzipiell wie folgt aufgeschlüsselt.

Projektleitung: Kindle Tristan und Ridvan Sevimli (Stellvertretung).

Dokumentation: Grundsätzlich alle, wobei Kindle Tristan, Waldburger Safiyya und Walser Christoph Marjan Markus für die Aufbereitung und Finalisierung zuständig sind.

Überwachung GitHub Funktionalitäten: Guerotto Sandro und Ridvan Sevimli.

Aktualisierung GitHub *issues*: Alle für die ihnen zugewiesenen *issues*.

Implementation: Grundsätzlich alle, wobei folgende Zweierteams definiert wurden.

- **Anbindung an Cloud-Dienste:** Guerotto Sandro und Waldburger Safiyya.
- **Dokumentenverwaltung (File-Browser):** Sevimli Ridvan und Walser Christoph Marjan Markus.
- **Backend (inkl. Verschlüsselung):** Tristan Kindle und Sieber Lorenz.

Alle Mitglieder haben bis zum Stand der Abgabe gleichermassen zum Projekt beigetragen (die obige Aufzählung soll lediglich die jeweiligen Schwerpunkte hervorheben).

10. Selbstständigkeitserklärung

Der vorliegende Bericht wurde von den genannten Personen (siehe Ziffer 9.1) selbstständig erarbeitet. Es wurden keine anderen als die angegebenen Hilfsmittel und Quellen benutzt. Dies gilt insbesondere für Informationen aus dem Internet. Alle sinngemäss und wörtlich übernommenen Textstellen aus fremden Quellen wurden kenntlich gemacht.

Zürich, 20. April 2022

Gruppe 4, Pretty Secure Cloud

11. Glossar

Cipher:	Methode oder Algorithmus für Ver- und Entschlüsselung
Continuous Integration:	“Continuous Integration” (im Deutschen auch als „kontinuierliche Integration“ oder „permanente Integration“ bezeichnet) ist eine Technik der agilen Softwareentwicklung. Bei dieser Art der Integration fügen Entwickler fertige Code-Schnipsel regelmäßig – mitunter mehrfach am Tag – in die Anwendung ein, statt sie alle erst zum Abschluss des Projekts zu integrieren [2].
Download:	Herunterladen bzw. Empfangen von Daten von einem bestimmten Medium
FURPS:	Akronym, welches sich auf die englischen Bezeichnungen für ein Qualitätsmodell hinsichtlich Software-Lebenszyklus bezieht [3].
FXML:	XML basierte Markup-Language, spezialisiert, um JavaFX GUIs zu erstellen.
GUI:	Graphical User Interface, Benutzeroberfläche
KISS-Prinzip:	Keep it simple and stupid: Möglichst einfache Lösung für ein Problem
Look-and-Feel	bezeichnet audiovisuelle Design-Aspekte von Software mit grafischer Benutzeroberfläche
MVC Pattern:	Model View Controller Pattern: Design Pattern für die GUI-Entwicklung
Pull-Anfrage:	Der Begriff "Pull-Anfrage" stammt von git, wobei die <pull Befehl> wird verwendet, um ein anderes Repository mit Ihrem lokalen Repository zusammenzuführen.
QuellCode:	Quellcode versteht man einen für Menschen lesbaren Text, der in einer bestimmten Programmiersprache verfasst ist [4].
Repository:	Als „Software-Repository“ versteht man ein „Projektarchiv“, in dem Quellcode für eine gewünschte Software entwickelt und versioniert wird [5].
Scenebuilder:	grafisches Tool zur Erstellung einer FXML-Datei
Sequenzdiagramm:	stellt die relevantesten Systemoperationen, die das System für die Anwendungsfälle (Use Cases) zur Verfügung stellen muss, dar
Testen:	Testen beschreibt den Prozess des Testens auf Funktionalität des bereits bestehenden Codes aber im allgemein meist einen Code Abschnitt der neu zu einem Quelltext hinzugefügt werden soll [6].
Three-tier-architecture:	Architekturprinzip in der Softwareentwicklung
UI:	User Interface, Benutzeroberfläche
UI-Sketch:	Benutzeroberfläche als Skizze
UML – Die Unified Modeling Language:	ist eine grafische Modellierungssprache zur Spezifikation, Konstruktion, Dokumentation und Visualisierung von (u.a.) Software.

Upload:	Hochladen bzw. Übertragen von Daten auf ein Medium
Use Case Diagramm:	Diagramm, welches die für die Software relevantesten Anwendungsfälle grafisch darlegt
Use Case:	Anwendungsfall oder Szenario, welches eine Interaktion des Akteurs mit dem System beschreibt
User:	Person, die das Programm verwendet, gleichbedeutend mit der/die Benutzer/-in
Versionskontrolle:	Die Versionskontrolle, auch bekannt als Quellcodekontrolle, ist ein Verfahren zur Verfolgung und Verwaltung von Änderungen an Softwarecode [7].
XML:	Extensible Markup Language: Definiert Regeln, um sowohl Maschinen- und Menschenlesbar zu sein.

12. Quellenverzeichnis

[1] E.Gamma, R.Helm, R.Johnson und J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1995, S. 107 - 116

[2] Web-Entwicklung (07.01.2022). *Was ist Continuous Integration* [Online].

<https://www.ionos.de/digitalguide/websites/web-entwicklung/continuous-integration/> [Stand 4.19.2022]

[3] (07.12.2009). *funcionality, usability, reliability, performance, supportability – FURPS* [Online].

<https://www.itwissen.info/funcionality-usability-reliability-performance-supportability-FURPS.html>

[Stand 19.4.2022]

[4] Web-Entwicklung (28.07.2020). *Definition Quellcode* [Online].

<https://www.ionos.de/digitalguide/websites/web-entwicklung/quellcode/> [Stand 19.4.2022]

[5] K. Pfeifer (24.04.2019). *Was ist ein Repository?* [Online].

<https://www.devguide.at/git/was-ist-ein-repository/> [Stand 19.4.2022]

[6] T. Klein und E. Semenova (21.01.2022). *Tests in der Softwareentwicklung: Ein Klassifizierungsansatz* [Online].

<https://www.redbots.de/blog/software-tests-klassifizierung/> [Stand 19.4.2022]

[7] T. Ebert (01.04.2020). *Was ist Versionskontrolle?* [Online].

<https://www.atlassian.com/de/git/tutorials/what-is-version-control> [Stand 19.4.2022]

13. Abbildungsverzeichnis

Abbildung 1 Use-Case-Diagramm	1
Abbildung 2 UI-Sketch Verschlüsselung und Upload	3
Abbildung 3 Sequenzdiagramm Verschlüsselung und Upload	4
Abbildung 4 UI-Sketch Download und Entschlüsselung	6
Abbildung 5 Sequenzdiagramm: Download und Entschlüsselung	7
Abbildung 6 UI-Sketch Registrierung eines neuen Accounts	8
Abbildung 7 Domänenmodell Pretty Secure Cloud	10
Abbildung 8 Package-Diagramm	11
Abbildung 9 Class-Diagramm	13
Abbildung 10 Beispiel: Registrierungsformular	14

14. Tabellenverzeichnis

Tabelle 1 Use-Case Verschlüsselung und Upload	2
Tabelle 2 Use-Case Download und Entschlüsselung	5
Tabelle 3 Use-Case Registrierung eines neuen Accounts	8

Tabelle 4 Farbpalette	14
Tabelle 5 Projektplan.....	16