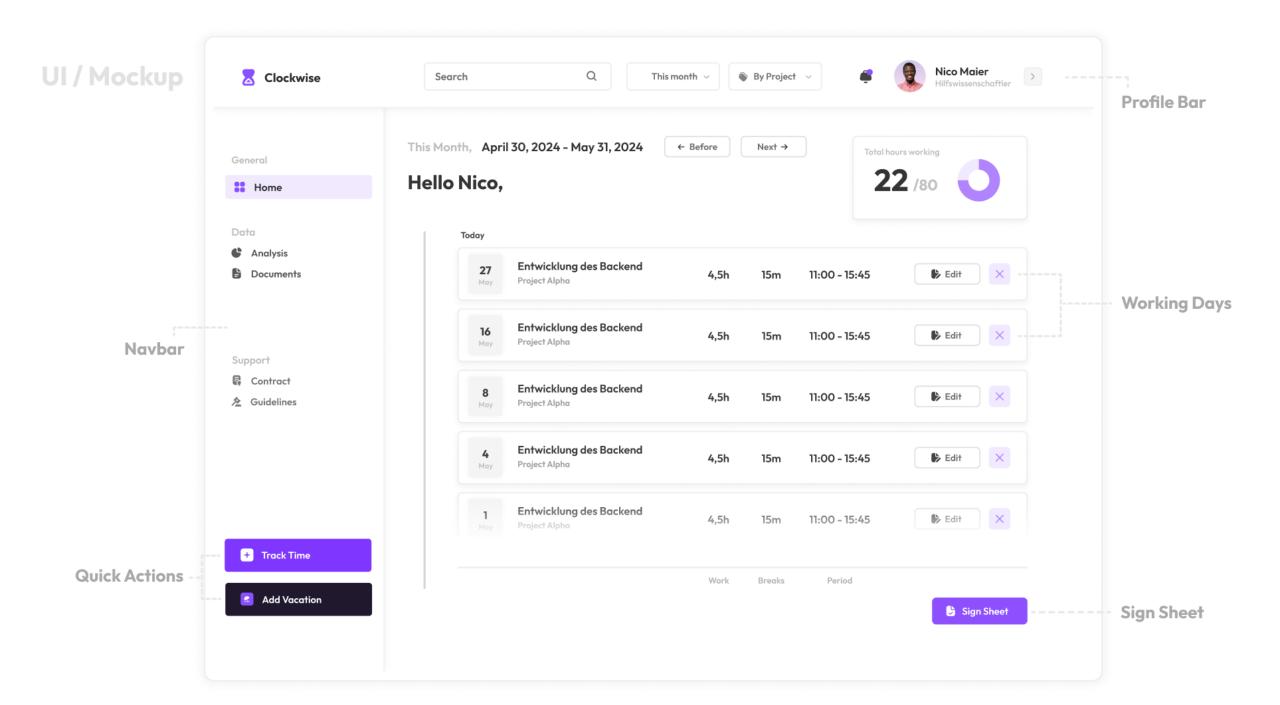


Intuitive Time Tracking

Team

Dominik Pollok Phil Gengenbach Alina Petri José Ayala Johann Kohl







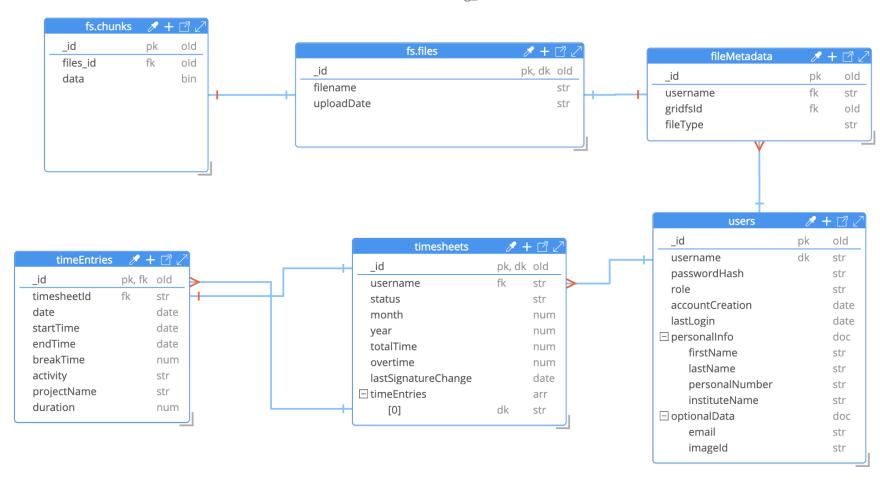
- 1. Datenbankentwurf
- 2. MVCS-Architektur
 - 2.1 MVCS am Beispiel User
- 3. Klassen Entwurf (UML)
 - 3.1 User Factory & Domänenklassen
 - 3.2 Timesheet & Time Entry
 - 3.3 Dateiverwaltung
- 4. Beispielablauf (Sequenzdiagramm)

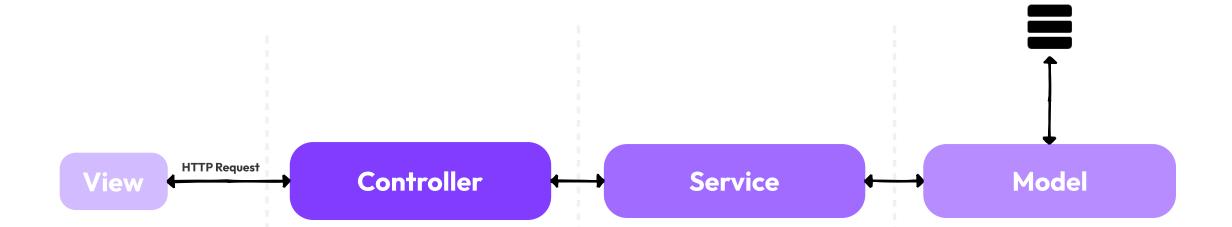


Datenbankentwurf









View Layer

- Darstellung der Benutzeroberfläche
- Nutzerinteraktion

Controller Layer

- Nimmt GET/POST/PATCH/DELETE Anfragen entgegen
- Weiterleitung an entsprechende Service Klassen

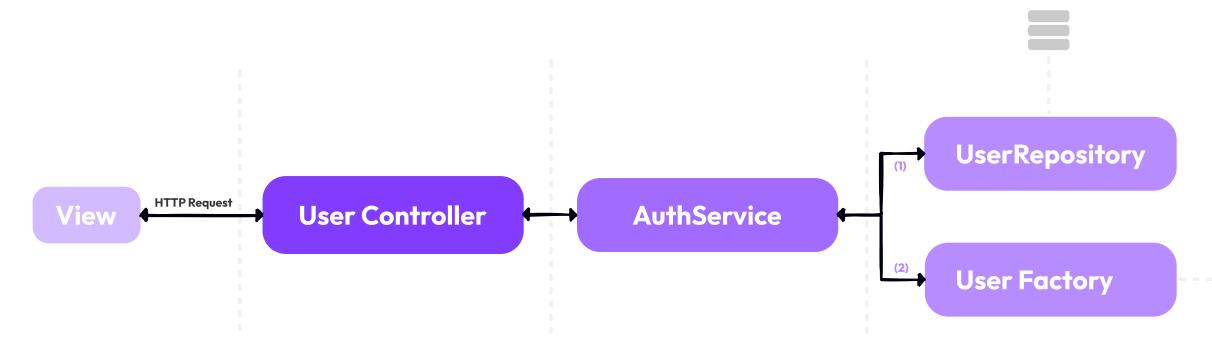
Service Layer

- Beinhaltet die Geschäftslogik
- Zugriff auf relevante Model-Klassen
- Validierung der Eingabedaten

Model Layer

- Repository-Klassen kommunizieren mit Datenbank
- Modellierung der unterschiedlichen Entitäten

Beispiel: Login Vorgang



Frontend

- Benutzeroberfläche
- Eingabe von Login Daten
- Daten in HTTP-Anfrage verpackt -> an Backend

Controller Layer

- Nimmt Login HTTP Anfrage von Frontend entgegen (JSON)
- Weiterleitung an Authentication Service Klasse

Service Layer

- Zuständig für Authentifizierung
- Überprüft Gültigkeit der Benutzerdaten -> Anfrage an UserRepository

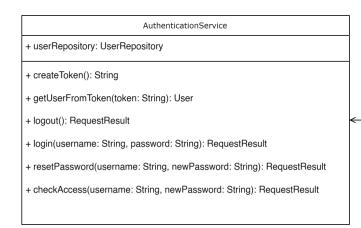
Model Layer

- User-Repo kommuniziert mit Datenbank (Gültigkeitsprüfung)
- User Factory zur Erstellung von User Objekten

User: Controller / Service

UML

- User Service: Verwaltung von Benutzerdaten (z.B.: Kontoerstellung, Nutzerlöschung)
- Authentication Service: Authentifizierung & Autorisierung (z.B.: Tokenerstellung, Zugriffkontrolle)



UserController + authenticationService: AuthenticationService + userService: UserService + fileService: FileService + createUser(): JSON + updateUser(): JSON + deleteUser(): JSON + login(): JSON + logout(): JSON + resetPassword(): JSON + getProfile(): JSON + getProfile(): JSON + getUsers(): JSON + getUsersByRole(): JSON + uploadUserFile(): JSON + getUserFile(): JSON + deleteUserFile(): JSON

UserService

+ userRepository: UserRepository

+ userValidator: InputValidator

+ createUser(userData: dict): RequestResult

+ updateUser(userData: dict): RequestResult

+ deleteUser(username: String): RequestResult

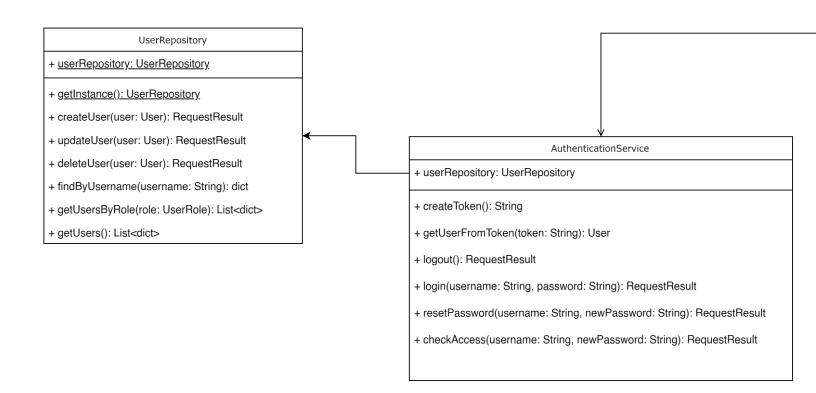
+ getUsers(): List<User>

+ getUsersByRole(role: String): List<User>

+ getProfile(username: String): User

User: Repository

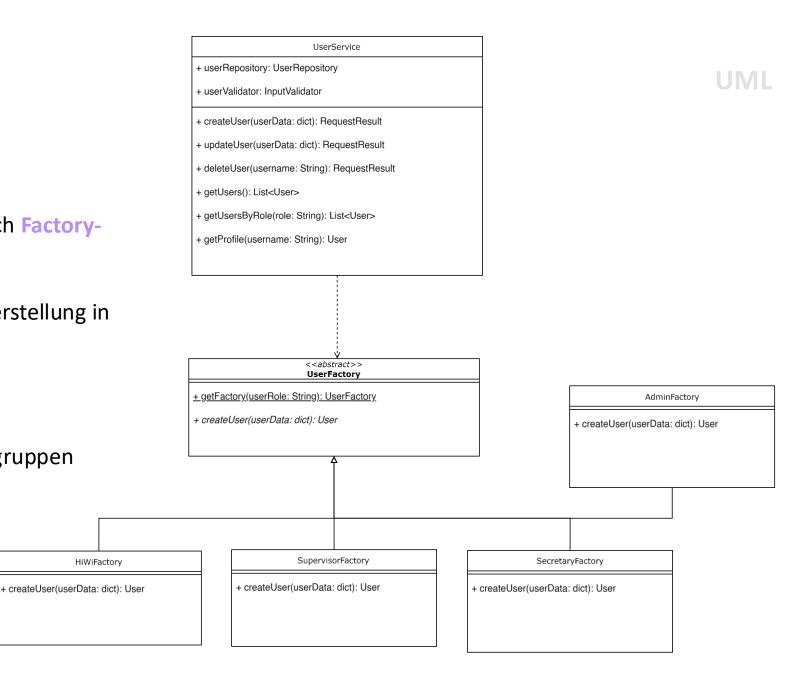
- Zentraler Bestandteil des Modells (für Datenbankzugriff zuständig)
- Singleton: nur eine Instanz
 -> Verwaltung der
 Datenbankverbindung effizienter & übersichtlicher
- Repository-Muster: abstrahiert
 Komplexität, klare Trennung von
 Datenbankzugriff und Geschäftslogik



User: Factory

- Erstellung von Nutzerobjekten nach Factory-**Method-Muster**
- Auslagerung der Logik zur Nutzererstellung in Factory Klassen
- Erhöhung der Wartbarkeit und Anpassungsfähigkeit z.B.: bei Integration neuer Nutzergruppen

HiWiFactory



User: Domänenklassen

- User Klasse im Zentrum: grundlegende Attribute, Rolle und persönliche Informationen
- Spezifischen Bedürfnisse der Nutzertypen durch Unterklassen modelliert z.B.: HiWi mit Vertragsinformationen oder Supervisor mit Liste an HiWis

ContractInfo

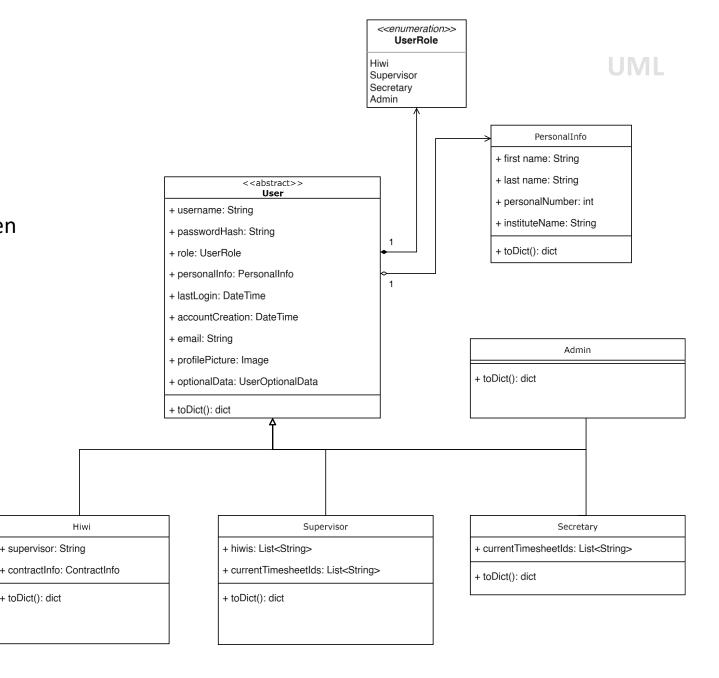
Hiwi

+ supervisor: String

+ toDict(): dict

+ hourlyWage: float + weeklyWorkingHours: int + vacationHours: int

+ toDict(): dict



Timesheet: Controller / Service

- Timesheet Service: Verwaltung von Timesheets
 z.B. Erstellen & Signieren von Timesheets
- Timesheet Controller: Schnittstelle zum Frontend

TimesheetController

- + timesheetService: TimesheetService
- + signTimesheet(): JSON
- + approveTimesheet(): JSON
- + requestChange(): JSON
- + getTimesheets(): JSON
- + getTimesheetByMonthYear(): JSON
- + getTimesheetByUsernameStatus(): JSON
- + getCurrentTimesheet(): JSON

TimesheetService

- + timesheetRepository: TimesheetRepository
- + timesheetValidator: TimesheetValidator
- + ensureTimesheetExists(username: String, month: int, year: int): RequestResult
- + signTimesheet(timsheetId: String): RequestResult
- + approveTimesheet(timesheetId: String): RequestResult
- + requestChange(timesheetId: String, description: String): RequestResult
- + setTimesheetStatus(timesheetId: String, newStatus:m TimesheetStatus): RequestResult
- + createTimesheet(username: String, month: int, year: int): RequestResult
- + getTimesheetByld(timesheetId: String): Timesheet
- + getTimesheetsByUsername(username: String): List<Timesheet>
- + getTimesheetsByUsernameStatus(username: String, status: TimesheetStatus): List<Timesheet>
- + getTimesheetId(username: String, month: int, year: int): String
- + addTimeEntryToTimesheet(timesheetId: String, newTimeEntryId: String): RequestResult
- + deleteTimeEntryFromTimesheet(timesheetId: String, timeEntryId: String): RequestResult
- + getTimesheet(username: String, month: int, year: int): Timesheet
- + getCurrentTimesheet(username: String): Timesheet

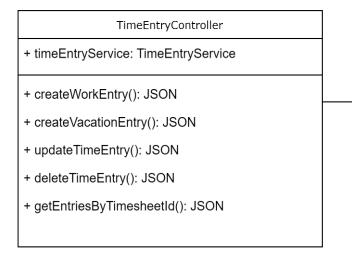
UML

Time Entry: Controller / Service

 Time Entry Service: Verwaltung von Time Entries

z.B.: Erstellen & Bearbeiten von Zeiteinträgen

 Time Entry Controller: Schnittstelle zum Frontend



+ timeEntryRepository: TimeEntryRepository

+ timesheetService: TimesheetService

+ entryValidator: TimeEntryDataValidator

+ createWorkEntry(entryData: dict): RequestResult

+ addVacationEntry(entryData: dict): RequestResult

+ updateTimeEntry(entryId: String, entryData: dict): RequestResult

+ deleteTimeEntry(timeEntryId: String): RequestResult

+ getEntriesOfTimesheet(timesheetId: String): List<TimeEntry>

TimeEntryService

Timesheet / Time Entry: Repository

- Datenbankzugriff zur Persistierung von Timesheets und Time Entries
- Singleton & Repository-Muster

TimeEntryRepository

timeEntryRepository: TimeEntryRepository

- + getInstance(): TimeEntryRepository
- + getTimeEntryById(timeEntryId: String): dict
- + getTimeEntriesByTimesheetId(timesheetId: String): List<dict>
- $+ \ update Time Entry (time Entry: Time Entry): Request Result \\$
- + deleteTimeEntry(entryId: String): RequestResult
- + createTimeEntry(timeEntry: TimeEntry): RequestResult

UML

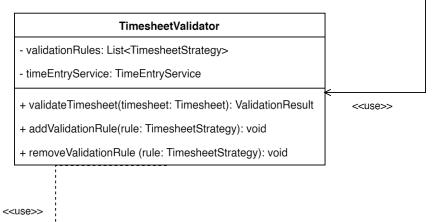
TimesheetRepository

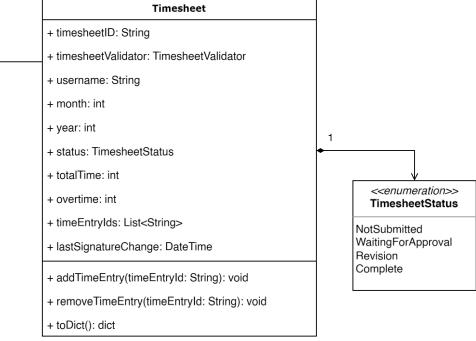
- timesheetRepository: TimesheetRespository
- + getInstance(): TimesheetRepository
- + getTimesheetById(timesheetId: String): dict
- + getTimesheet(username: String, month: int, year: int): dict
- + getCurrentTimesheet(username: String): dict
- + getTimesheetsByTimePeriod(username: String, startDate: Date, endDate: Date): List<dict>
- + getTimesheets(username: String): List<dict>
- + getTimesheetsByStatus(status: TimesheetStatus): List<dict>
- + getTimesheetId(username: String, month: int, year: int): String
- + updateTimesheet(timesheet: Timesheet): RequestResult
- + setTimesheetStatus(timesheetId: String, newStatus: TimesheetStatus): RequestResult
- + deleteTimesheetById(timesheetId: String): RequestResult
- + createTimesheet(timesheet: Timesheet): RequestResult
- + getTimesheetsByUsernameStatus(username: String, status: TimesheetStatus): List<dict>

Timesheet: Domänenklassen

UML

- Timesheet: grundlegende Attribute,
 Timesheet Validator und Liste an Time Entries
- Timesheet Validator: validiert Timesheet & legt
 Validierungsstrategien fest

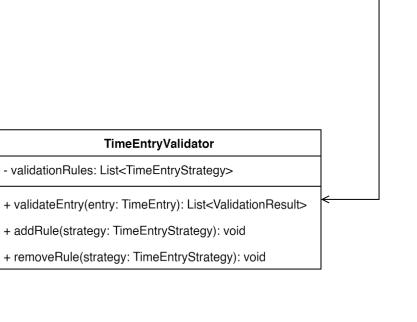




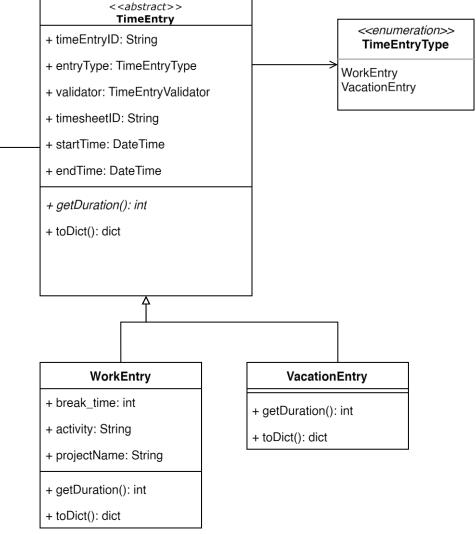
Time Entry: Domänenklassen

Time Entry: grundlegende Attribute,
 Time Entry Type, Time Entry Validator

 Time Entry Validator: validiert Time Entries & legt Validierungsstrategien fest



UML



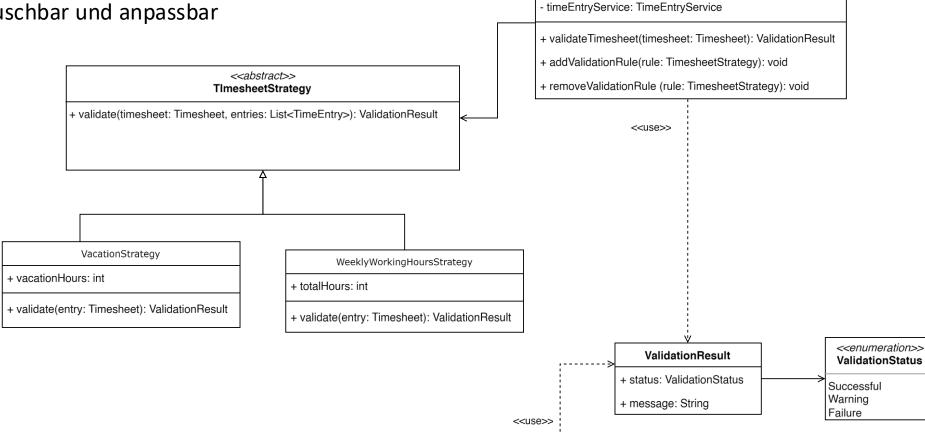
TimesheetValidator

validationRules: List<TimesheetStrategy>

Timesheet: Validierung

Timesheet Validator: validiert Timesheet bei Signatur

 Strategy-Pattern: Validierungs-Strategien einfach erweiterbar, austauschbar und anpassbar



Time Entry: Validierung

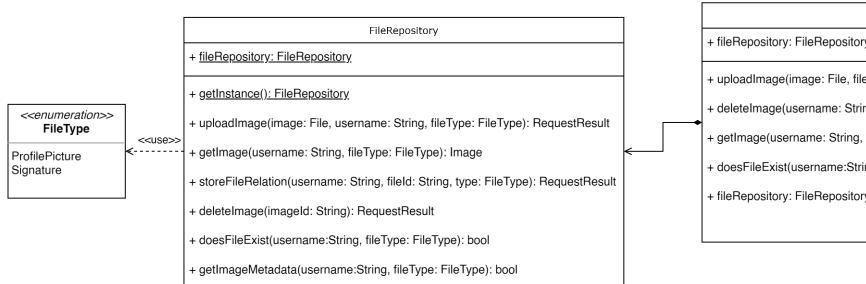
Time Entry Validator: validiert Zeiteinträge <<enumeration>> ValidationResult **ValidationStatus** + status: ValidationStatus Successful **Strategy-Pattern:** Validierungs-Strategien einfach Warning + message: String Failure erweiterbar, austauschbar und anpassbar <<use>>> **TimeEntryValidator** <<abstract>> TimeEntryStrategy validationRules: List<TimeEntryStrategy> + validate(entry: TimeEntry): ValidationResult + validateEntry(entry: TimeEntry): List<ValidationResult> + addRule(strategy: TimeEntryStrategy): void + removeRule(strategy: TimeEntryStrategy): void BreakLengthStrategy WorkingTimeStrategy HolidayStrategy + minBreakLength: int + workingStart: int + holidays: HolidayCalendar + workingEnd: int + validate(entry: TimeEntry): ValidationResult + validate(entry: TimeEntry): ValidationResult + maxDailyHours: int + validate(entry: TimeEntry): ValidationResult HolidayCalendar + holidays: List<Date>

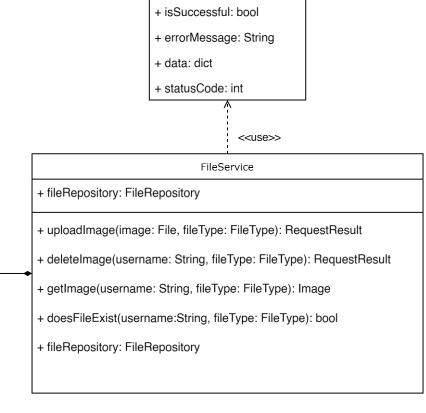
+ isHoliday(date): bool

Dateiverwaltung

 FileService: Verantwortlich für sämtliche Dateioperationen, integriert Geschäftslogik und erforderliche Validierungen

 RequestResult: Ergebnis einer Operation, handhabt Fehler und steuert Prozessabläufe





RequestResult

Document: Controller/Service

- Document Controller: Steuerung und Koordination der Dokumentenerstellung
- Document Service: Erzeugung, Verwaltung und Verarbeitung von Dokumenten.
- Anbindung von Document Service an Strategy-Pattern

+ documentService: DocumentService + fileService: FileService + generateDocument(): JSON + generateMultipleDocuments(): JSON

DocumentService

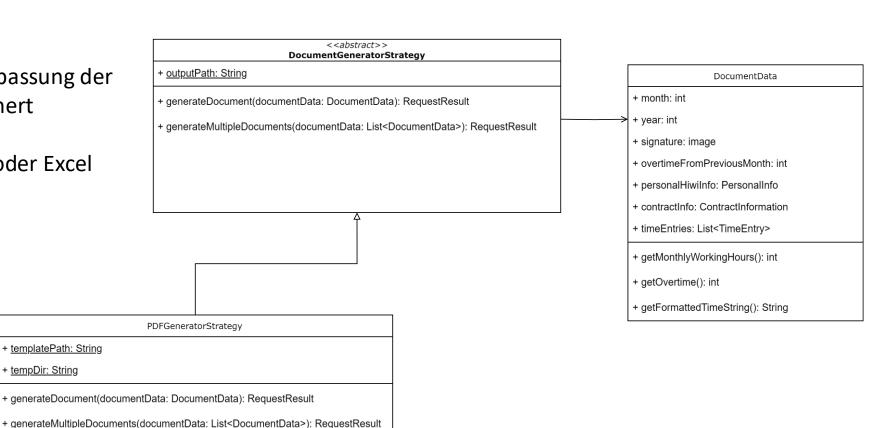
- selectedDocumentGenerator: DocumentGenerator
- fileService: FileService
- timesheetService: TimesheetService
- timeEntryService: TimeEntryService
- + gatherDocumentData(username: String, month: int, year: int): DocumentData
- + generateDocument(username: String, month: int, year: int): RequestResult
- + generateMultipleDocuments(usernames: List<String>, month: int, year:int): RequestResult
- + generateMultipleDocumentsById(timesheetIds: List<String>): RequestResult
- + generateDocumentsInDateRange(username: String, startDate: Date, endDate: Date): RequestResult

UML

+ templatePath: String + tempDir: String

UML

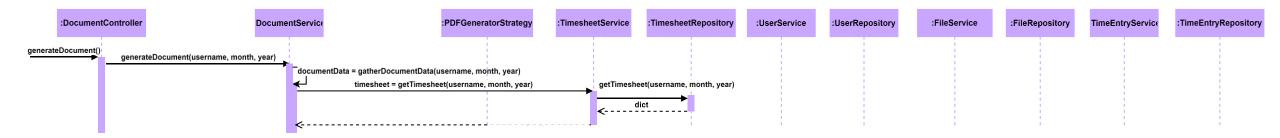
Strategy-Pattern: flexible Anpassung der Algorithmen zur Laufzeit, sichert Erweiterbarkeit z.B.: bei Integration von .txt oder Excel **Format**

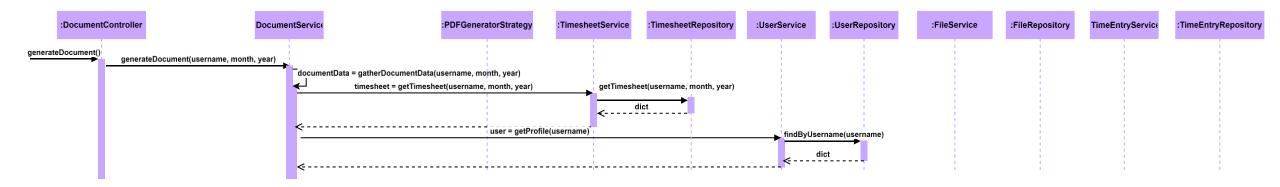


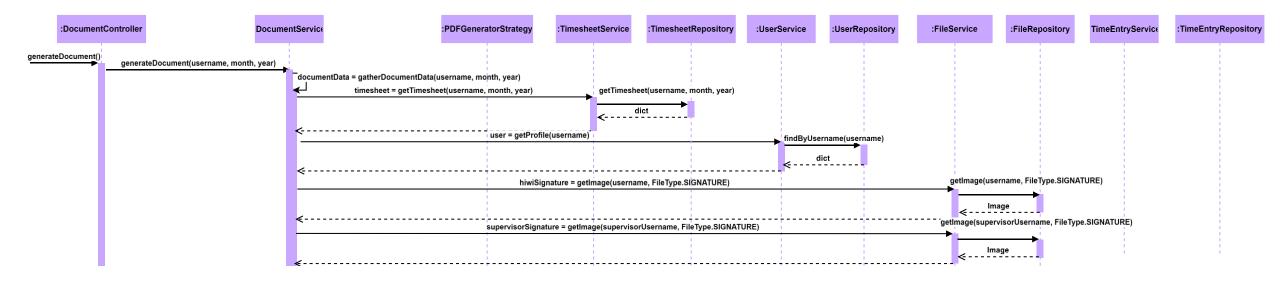
Sequenzdiagramm

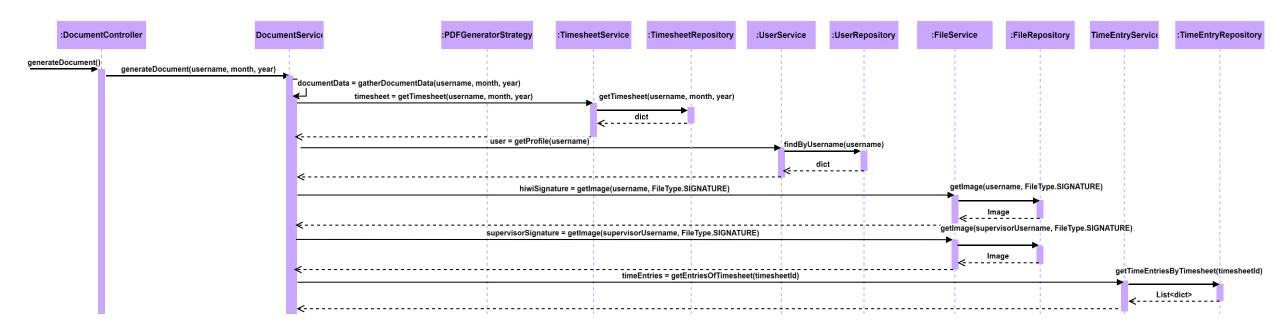
:DocumentController :TimesheetService :TimesheetRepository :UserService :TimeEntryRepository :TimeEntryRepository

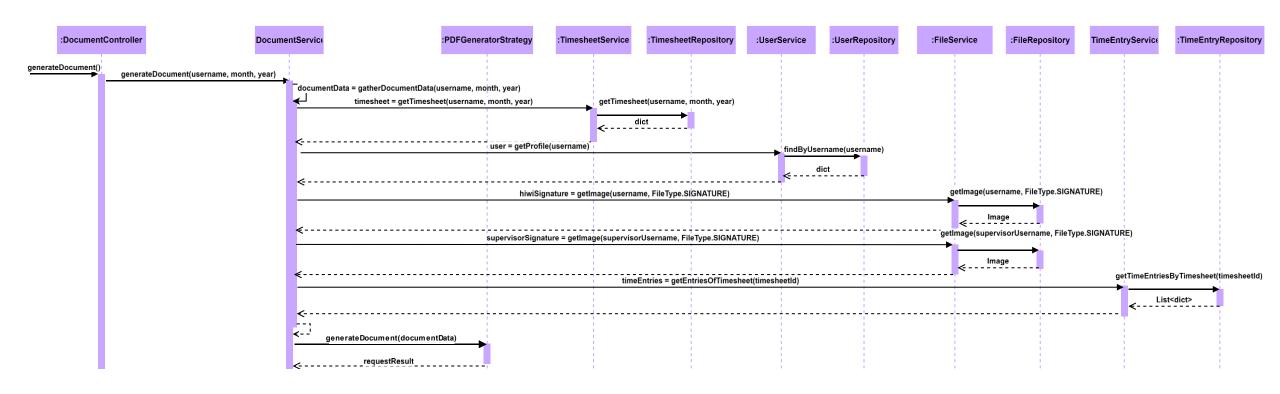


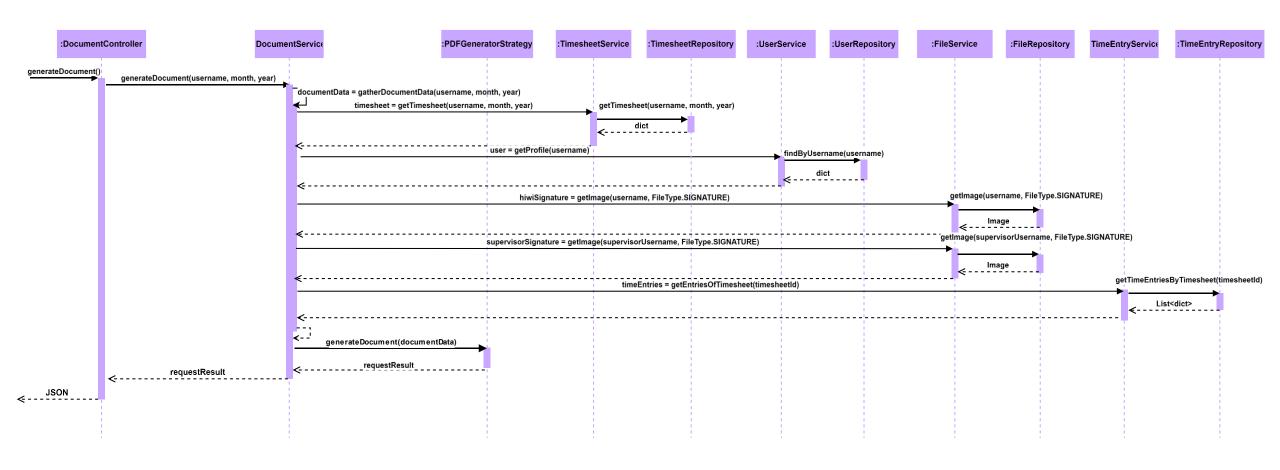


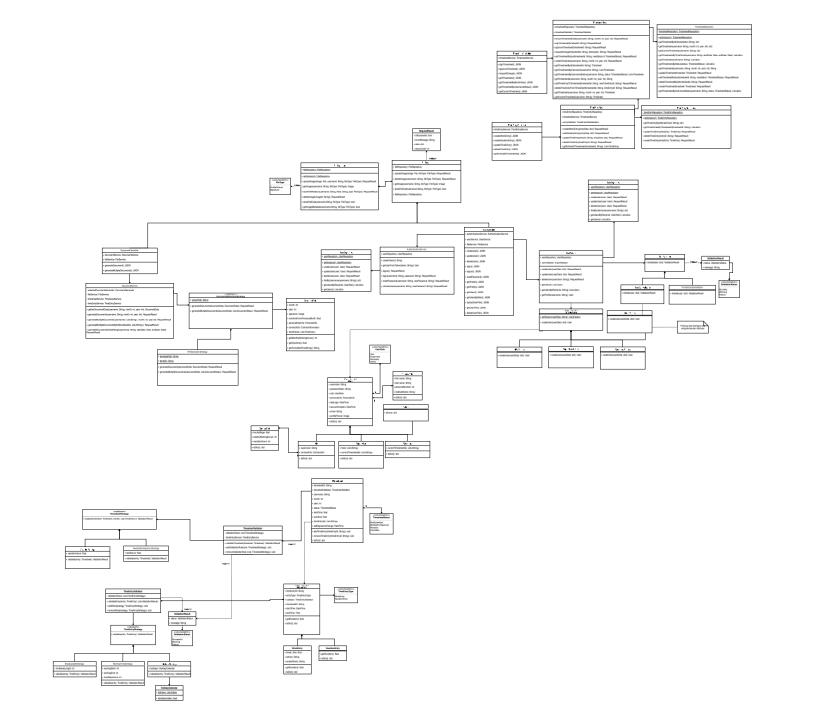












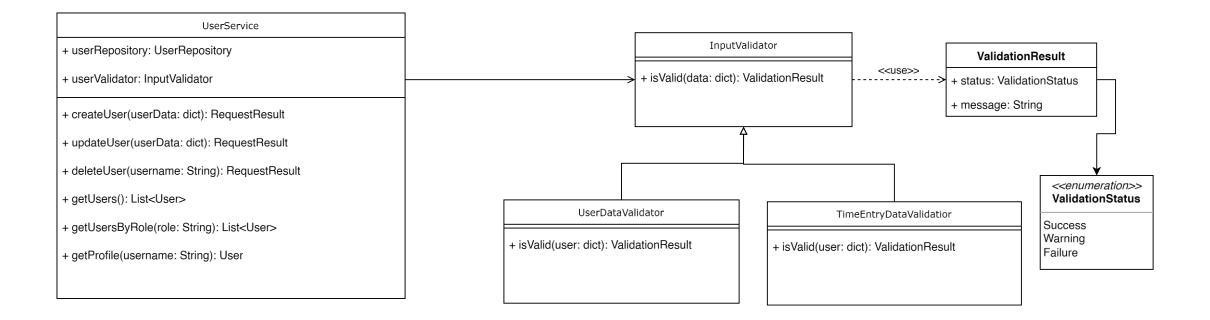




Team

Dominik Pollok Phil Gengenbach Alina Petri José Ayala Johann Kohl





Timesheet MVCS Fragen

TimesheetController

- + timesheetService: TimesheetService
- + signTimesheet(): JSON
- + approveTimesheet(): JSON
- + requestChange(): JSON
- + getTimesheets(): JSON
- + getTimesheetByMonthYear(): JSON
- + getTimesheetByUsernameStatus(): JSON
- + getCurrentTimesheet(): JSON

TimesheetService

- + timesheetRepository: TimesheetRepository
- + timesheetValidator: TimesheetValidator
- + ensureTimesheetExists(username: String, month: int, year: int): RequestResult
- + signTimesheet(timsheetId: String): RequestResult
- + approveTimesheet(timesheetId: String): RequestResult
- + requestChange(timesheetId: String, description: String): RequestResult
- + setTimesheetStatus(timesheetId: String, newStatus:m TimesheetStatus): RequestResult
- + createTimesheet(username: String, month: int, year: int): RequestResult
- + getTimesheetByld(timesheetId: String): Timesheet
- + getTimesheetsByUsername(username: String): List<Timesheet>
- + getTimesheetsByUsernameStatus(username: String, status: TimesheetStatus): List<Timesheet>
- + getTimesheetId(username: String, month: int, year: int): String
- + addTimeEntryToTimesheet(timesheetId: String, newTimeEntryId: String): RequestResult
- + deleteTimeEntryFromTimesheet(timesheetId: String, timeEntryId: String): RequestResult
- + getTimesheet(username: String, month: int, year: int): Timesheet
- + getCurrentTimesheet(username: String): Timesheet

TimesheetRepository

- timesheetRepository: TimesheetRespository
- + getInstance(): TimesheetRepository
- + getTimesheetById(timesheetId: String): dict
- + getTimesheet(username: String, month: int, year: int): dict
- getCurrentTimesheet(username: String): dict
- + getTimesheetsByTimePeriod(username: String, startDate: Date, endDate: Date): List<dict>
- + getTimesheets(username: String): List<dict>
- + getTimesheetsByStatus(status: TimesheetStatus): List<dict>
- + getTimesheetId(username: String, month: int, year: int): String
- + updateTimesheet(timesheet: Timesheet): RequestResult
- + setTimesheetStatus(timesheetId: String, newStatus: TimesheetStatus): RequestResult
- + deleteTimesheetByld(timesheetId: String): RequestResult
- + createTimesheet(timesheet: Timesheet): RequestResult
- + getTimesheetsByUsernameStatus(username: String, status: TimesheetStatus): List<dict>