Arbeitszeitdokumentation

Von Michael Große, Judith Neugebauer und Philipp Leschke

Kontext

3-Schichten

Das Drei-Schichten-Modell ist eine Software-Architektur, die drei unterschiedliche Schichten - die Schicht der Benutzeroberfläche, die Fachkonzeptschicht und eine Datenhaltungsschicht - unterscheidet. Die Schicht der Benutzeroberfläche bildet die Funktionalität einer grafischen oder textuellen Benutzeroberfläche ab. Auf dieser Schicht wäre auch die Implementation einer API denkbar. Die Aufgabe der Fachkonzeptschicht ist die Realisierung der eigentlichen fachlichen Anforderungen und der entsprechenden Schnittstelle, um auf die darunterliegende Datenhaltungsschicht zugreifen zu können. Die Datenhaltungsschicht trägt die Verantwortung für das Speichern und Laden der Daten. Die Schichten sind durch Interfaces lediglich lose gekoppelt. Die Fachkonzeptschicht und die Datenhaltungsschicht können also nicht auf die Schicht der Benutzeroberfläche zugreifen. Keine der beiden Schichten hat explizites Wissen über die Benutzeroberfläche. Umgekehrt hat das UI a priori kein spezifischens Wissen über die darunter liegenden Schichten. Die Software wird in 3-Schichtenarchitektur implementiert um auf geänderte Anforderungen flexibel reagieren zu können. Ein Teil der Funktionalität der Fachkonzeptschicht wird mit Hilfe des "Decorator" Design-Patterns umgesetzt.

Das Decorator Entwurfsmuster

Der Zweck des Decorator-Musters ist es eine Klasse um Funktionalität zu erweitern bzw. verändern ohne Unterklassen zu bilden und die öffentliche API der Klasse beizubehalten. Auf diese Art und Weise können voneinander unabhängige Funktionalitäten zu einer Klasse hinzugefügt werden ohne für jede Kombination Unterklassen bilden zu müssen.

Im Allgemeinen besteht diese Klasse aus einer abstrakten Decorator-Basisklasse welche von der zu erweiternden Klasse bzw. Interface erbt und einer Anzahl von konkreten Decorator-Klasses welche von dieser Decorator-Basisklasse erben. Die Basisklasse erhält im Konstruktor eine Instanz der zu erweiternden Klasse und leitet alle Methodenaufrufe an diese Klasse weiter und gibt die entsprechenden Rückgabewerte selbst unverändert zurück. Konkrete Dekoratoren überschreiben dann nur die für sie relevanten Methoden.

Ein verbreitetes Beispiel wäre ein Objekt des GUI welches um einen Rahmen und eine Scrollbar erweitert werden soll. Ohne das Decorator-Muster müssten dafür 4 Unterklassen gebildet werden: Für das Objekt nur mit Scrollbar, für das Objekt mit Scrollbar innerhalb des Rahmens, für das Objekt mit Scrollbar außerhalb des Rahmens und für das Objekt nur mit Rahmen. Mit dem Decorator-Muster lässt sich das mit einem Decorator für "Zeichne einen Rahmen um das Objekt" und "Zeichen eine Scrollbar an das Objekt" lösen.

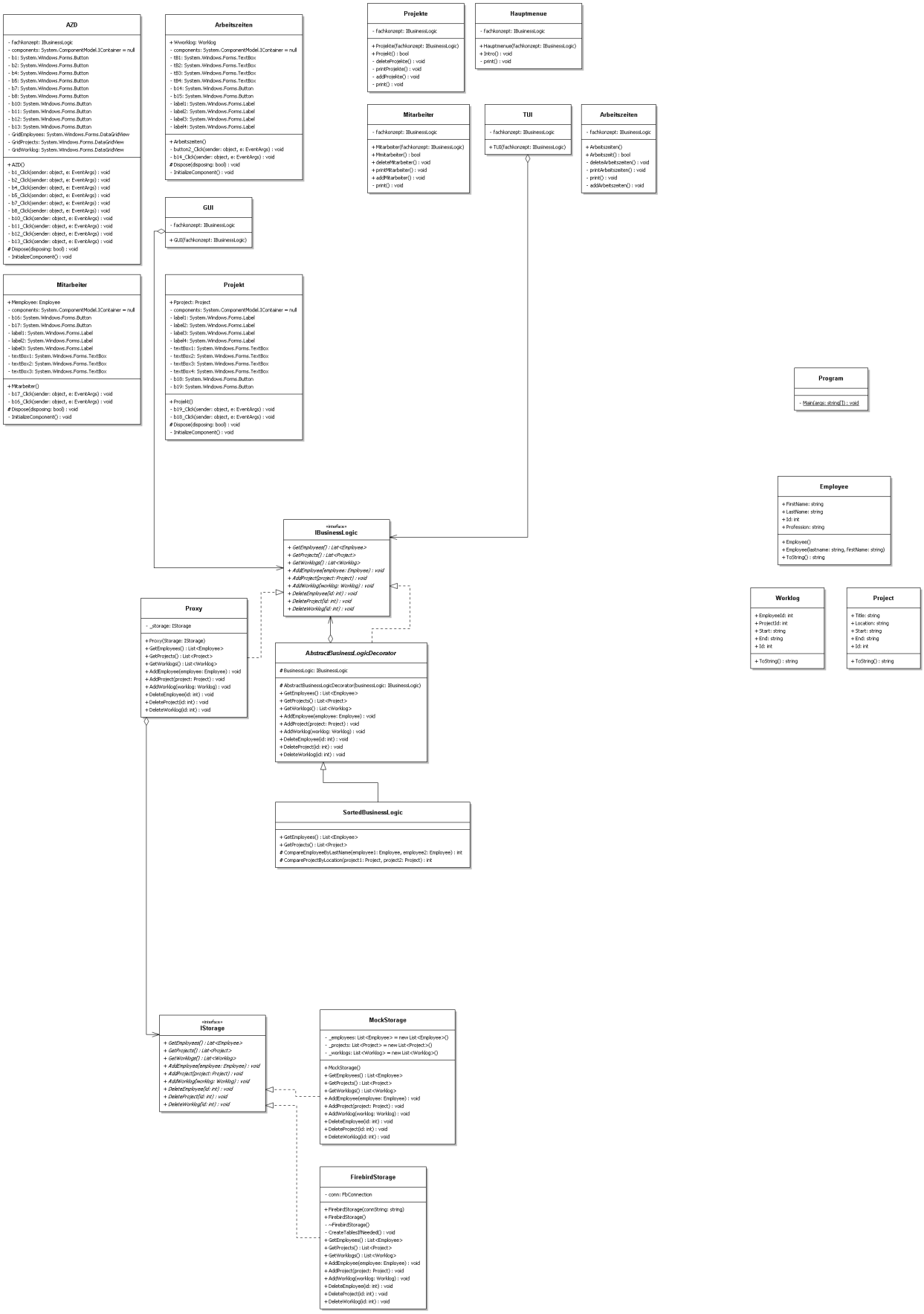
Aufgabe der Software

Der Kunde benötigt eine Software um die Arbeitszeiten der Mitarbeiter an bestimmten Projekten abzuspeichern und anzeigen zu lassen.

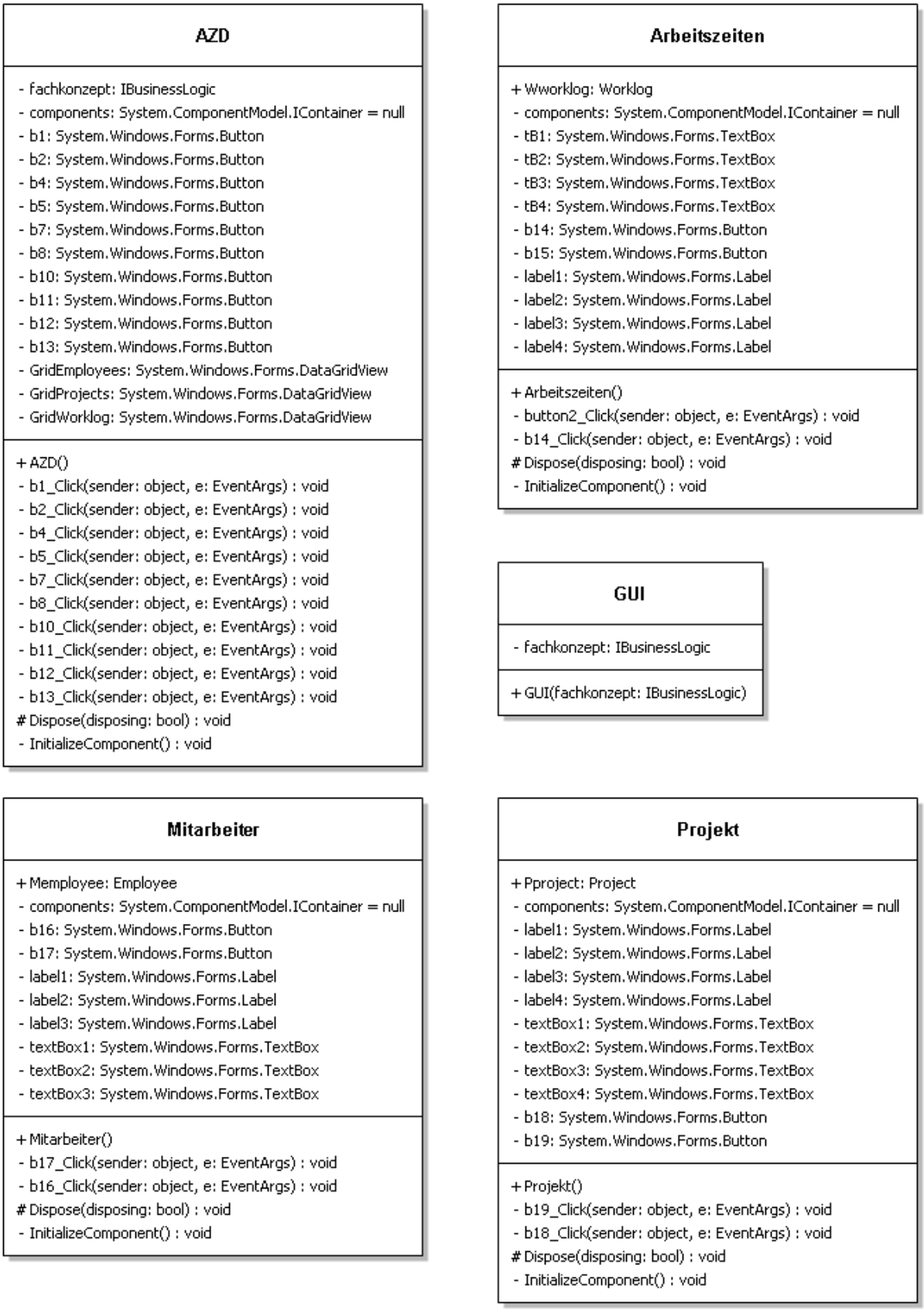
konkretes Produkt

Architektur

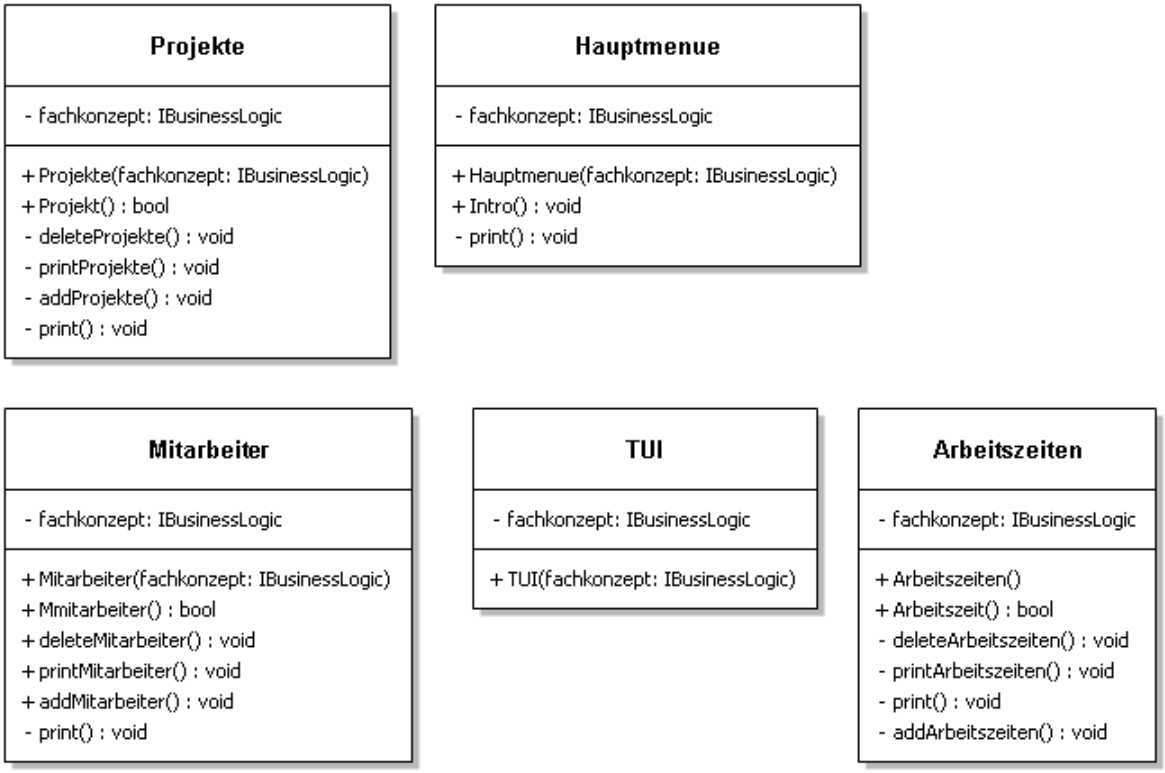
Übersicht


AZD
- fachkonzept: IBusinessLogic
- components: System.ComponentModel.IContainer = null
- b1: System.Windows.Forms.Button
- b2: System.Windows.Forms.Button
- b4: System.Windows.Forms.Button
- b5: System.Windows.Forms.Button
- b7: System.Windows.Forms.Button
- b8: System.Windows.Forms.Button
- b10: System.Windows.Forms.Button
- b11: System.Windows.Forms.Button
- b12: System.Windows.Forms.Button
- b13: System.Windows.Forms.Button
- GridEmployees: System.Windows.Forms.DataGridView
- GridProjects: System.Windows.Forms.DataGridView
- GridWorklog: System.Windows.Forms.DataGridView
+ AZD()
- b1_Click(sender: object, e: EventArgs) : void
- b2_Click(sender: object, e: EventArgs) : void
- b4_Click(sender: object, e: EventArgs) : void
- b5_Click(sender: object, e: EventArgs) : void
- b7_Click(sender: object, e: EventArgs) : void
- b8_Click(sender: object, e: EventArgs) : void
- b10_Click(sender: object, e: EventArgs) : void
- b11_Click(sender: object, e: EventArgs) : void
- b12_Click(sender: object, e: EventArgs) : void
- b13_Click(sender: object, e: EventArgs) : void
Dispose(disposing: bool) : void
- InitializeComponent() : void
Arbeitszeiten
+ WworklogL Worklog
- components: System.ComponentModel.IContainer = null
- tB1: System.Windows.Forms.TextBox
- tB2: System.Windows.Forms.TextBox
- tB3: System.Windows.Forms.TextBox
- tB4: System.Windows.Forms.TextBox
- b14: System.Windows.Forms.Button
- b15: System.Windows.Forms.Button
- label1: System.Windows.Forms.Label
- label2: System.Windows.Forms.Label
- label3: System.Windows.Forms.Label
- label4: System.Windows.Forms.Label
+ Arbeitszeiten()
- button2_Click(sender: object, e: EventArgs) : void
- b14_Click(sender: object, e: EventArgs) : void
Dispose(disposing: bool) : void
- InitializeComponent() : void
Projekte
- fachkonzept: IBusinessLogic
+ Projekte(fachkonzept: IBusinessLogic)
+ ProjekE() : bool
- deleteProjekte() : void
- printProjekte() : void
- addProjekte() : void
- print() : void
Hauptmenue
- fachkonzept: IBusinessLogic
+ Hauptmenue(fachkonzept: IBusinessLogic)
+ Intro() : void
- print() : void
Mitarbeiter
- fachkonzept: IBusinessLogic
+ Mitarbeiter(fachkonzept: IBusinessLogic)
+ MmitarbeiterE() : bool
+ deleteMitarbeiter() : void
+ printMitarbeiter() : void
+ addMitarbeiter() : void
- print() : void
TUI
- fachkonzept: IBusinessLogic
+ TUI(fachkonzept: IBusinessLogic)
Arbeitszeiten
- fachkonzept: IBusinessLogic
+ Arbeitszeiten()
+ ArbeitszeitE() : bool
- deleteArbeitszeiten() : void
- printArbeitszeiten() : void
- print() : void
- addArbeitszeiten() : void
GUI
- fachkonzept: IBusinessLogic
+ GUI(fachkonzept: IBusinessLogic)
Mitarbeiter
+ Memployee: Employee
- components: System.ComponentModel.IContainer = null
- b16: System.Windows.Forms.Button
- b17: System.Windows.Forms.Button
- label1: System.Windows.Forms.Label
- label2: System.Windows.Forms.Label
- label3: System.Windows.Forms.Label
- textBox1: System.Windows.Forms.TextBox
- textBox2: System.Windows.Forms.TextBox
- textBox3: System.Windows.Forms.TextBox
+ Mitarbeiter()
- b17_Click(sender: object, e: EventArgs) : void
- b16_Click(sender: object, e: EventArgs) : void
Dispose(disposing: bool) : void
- InitializeComponent() : void
Projekt
+ Pproject: Project
- components: System.ComponentModel.IContainer = null
- label1: System.Windows.Forms.Label
- label2: System.Windows.Forms.Label
- label3: System.Windows.Forms.Label
- label4: System.Windows.Forms.Label
- textBox1: System.Windows.Forms.TextBox
- textBox2: System.Windows.Forms.TextBox
- textBox3: System.Windows.Forms.TextBox
- textBox4: System.Windows.Forms.TextBox
- b18: System.Windows.Forms.Button
- b19: System.Windows.Forms.Button
+ ProjekE()
- b19_Click(sender: object, e: EventArgs) : void
- b18_Click(sender: object, e: EventArgs) : void
Dispose(disposing: bool) : void
- InitializeComponent() : void
Program
- Main(args: string[]) : void
Employee
+ FirstName: string
+ LastName: string
+ Id: int
+ Profession: string
+ Employee()
+ Employee(lastname: string, FristName: string)
+ ToString() : string
«interface»
IBusinessLogic
+ GetEmployees() : List<Employee>
+ GetProjects() : List<Project>
+ GetWorklogs() : List<Worklog>
+ AddEmployee(employee: Employee) : void
+ AddProject(project: Project) : void
+ AddWorklog(worklog: Worklog) : void
+ DeleteEmployee(id: int) : void
+ DeleteProject(id: int) : void
+ DeleteWorklog(id: int) : void
Worklog
+ EmployeeId: int
+ ProjectId: int
+ Start: string
+ End: string
+ Id: int
+ ToString() : string
Project
+ Title: string
+ Location: string
+ Start: string
+ End: string
+ Id: int
+ ToString() : string
Proxy
- _storage: IStorage
+ Proxy(Storage: IStorage)
+ GetEmployees() : List<Employee>
+ GetProjects() : List<Project>
+ GetWorklogs() : List<Worklog>
+ AddEmployee(employee: Employee) : void
+ AddProject(project: Project) : void
+ AddWorklog(worklog: Worklog) : void
+ DeleteEmployee(id: int) : void
+ DeleteProject(id: int) : void
+ DeleteWorklog(id: int) : void
AbstractBusinessLogicDecorator
BusinessLogic: IBusinessLogic
AbstractBusinessLogicDecorator(businessLogic: IBusinessLogic)
+ GetEmployees() : List<Employee>
+ GetProjects() : List<Project>
+ GetWorklogs() : List<Worklog>
+ AddEmployee(employee: Employee) : void
+ AddProject(project: Project) : void
+ AddWorklog(worklog: Worklog) : void
+ DeleteEmployee(id: int) : void
+ DeleteProject(id: int) : void
+ DeleteWorklog(id: int) : void
SortedBusinessLogic
+ GetEmployees() : List<Employee>
+ GetProjects() : List<Project>
# CompareEmployeeByLastName(employee1: Employee, employee2: Employee) : int
# CompareProjectByLocation(project1: Project, project2: Project) : int
«interface»
IStorage
+ GetEmployees() : List<Employee>
+ GetProjects() : List<Project>
+ GetWorklogs() : List<Worklog>
+ AddEmployee(employee: Employee) : void
+ AddProject(project: Project) : void
+ AddWorklog(worklog: Worklog) : void
+ DeleteEmployee(id: int) : void
+ DeleteProject(id: int) : void
+ DeleteWorklog(id: int) : void
MockStorage
- _employees: List<Employee> = new List<Employee>()
- _projects: List<Project> = new List<Project>()
- _worklogs: List<Worklog> = new List<Worklog>()
+ MockStorage()
+ GetEmployees() : List<Employee>
+ GetProjects() : List<Project>
+ GetWorklogs() : List<Worklog>
+ AddEmployee(employee: Employee) : void
+ AddProject(project: Project) : void
+ AddWorklog(worklog: Worklog) : void
+ DeleteEmployee(id: int) : void
+ DeleteProject(id: int) : void
+ DeleteWorklog(id: int) : void
FirebirdStorage
- conn: FbConnection
+ FirebirdStorage(connString: string)
+ FirebirdStorage()
- ~FirebirdStorage()
- CreateTablesIfNeeded() : void
+ GetEmployees() : List<Employee>
+ GetProjects() : List<Project>
+ GetWorklogs() : List<Worklog>
+ AddEmployee(employee: Employee) : void
+ AddProject(project: Project) : void
+ AddWorklog(worklog: Worklog) : void
+ DeleteEmployee(id: int) : void
+ DeleteProject(id: int) : void
+ DeleteWorklog(id: int) : void

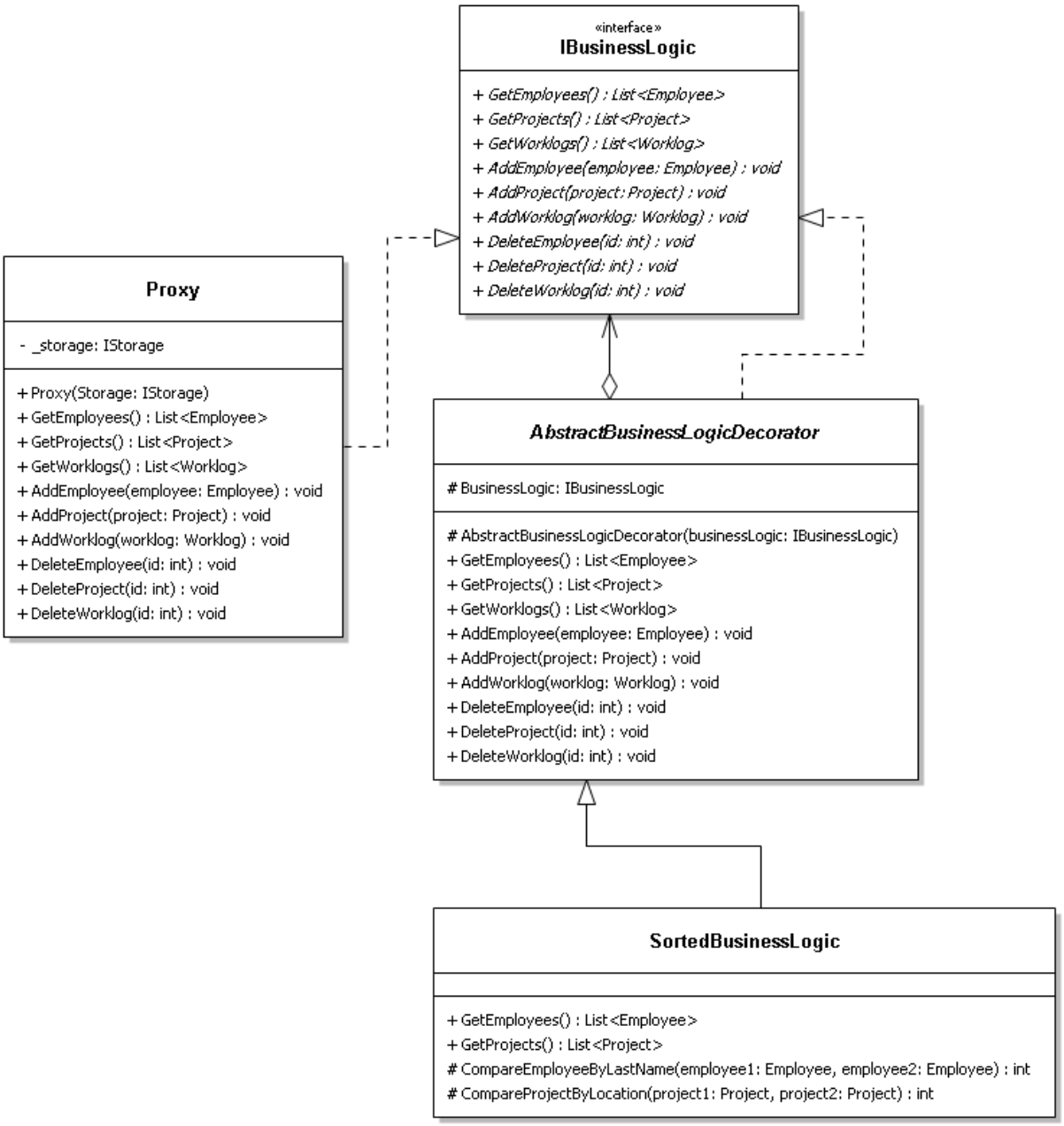
GUI


AZD
- fachkonzept: IBusinessLogic
- components: System.ComponentModel.IContainer = null
- b1: System.Windows.Forms.Button
- b2: System.Windows.Forms.Button
- b4: System.Windows.Forms.Button
- b5: System.Windows.Forms.Button
- b7: System.Windows.Forms.Button
- b8: System.Windows.Forms.Button
- b10: System.Windows.Forms.Button
- b11: System.Windows.Forms.Button
- b12: System.Windows.Forms.Button
- b13: System.Windows.Forms.Button
- GridEmployees: System.Windows.Forms.DataGridView
- GridProjects: System.Windows.Forms.DataGridView
- GridWorklog: System.Windows.Forms.DataGridView
+ AZD()
- b1_Click(sender: object, e: EventArgs) : void
- b2_Click(sender: object, e: EventArgs) : void
- b4_Click(sender: object, e: EventArgs) : void
- b5_Click(sender: object, e: EventArgs) : void
- b7_Click(sender: object, e: EventArgs) : void
- b8_Click(sender: object, e: EventArgs) : void
- b10_Click(sender: object, e: EventArgs) : void
- b11_Click(sender: object, e: EventArgs) : void
- b12_Click(sender: object, e: EventArgs) : void
- b13_Click(sender: object, e: EventArgs) : void
Dispose(disposing: bool) : void
- InitializeComponent() : void
Arbeitszeiten
+ Wworklog: Worklog
- components: System.ComponentModel.IContainer = null
- tB1: System.Windows.Forms.TextBox
- tB2: System.Windows.Forms.TextBox
- tB3: System.Windows.Forms.TextBox
- tB4: System.Windows.Forms.TextBox
- b14: System.Windows.Forms.Button
- b15: System.Windows.Forms.Button
- label1: System.Windows.Forms.Label
- label2: System.Windows.Forms.Label
- label3: System.Windows.Forms.Label
- label4: System.Windows.Forms.Label
+ Arbeitszeiten()
- button2_Click(sender: object, e: EventArgs) : void
- b14_Click(sender: object, e: EventArgs) : void
Dispose(disposing: bool) : void
- InitializeComponent() : void
GUI
- fachkonzept: IBusinessLogic
+ GUI(fachkonzept: IBusinessLogic)
Mitarbeiter
+ Memployee: Employee
- components: System.ComponentModel.IContainer = null
- b16: System.Windows.Forms.Button
- b17: System.Windows.Forms.Button
- label1: System.Windows.Forms.Label
- label2: System.Windows.Forms.Label
- label3: System.Windows.Forms.Label
- textBox1: System.Windows.Forms.TextBox
- textBox2: System.Windows.Forms.TextBox
- textBox3: System.Windows.Forms.TextBox
+ Mitarbeiter()
- b17_Click(sender: object, e: EventArgs) : void
- b16_Click(sender: object, e: EventArgs) : void
Dispose(disposing: bool) : void
- InitializeComponent() : void
Projekt
+ Pproject: Project
- components: System.ComponentModel.IContainer = null
- label1: System.Windows.Forms.Label
- label2: System.Windows.Forms.Label
- label3: System.Windows.Forms.Label
- label4: System.Windows.Forms.Label
- textBox1: System.Windows.Forms.TextBox
- textBox2: System.Windows.Forms.TextBox
- textBox3: System.Windows.Forms.TextBox
- textBox4: System.Windows.Forms.TextBox
- b18: System.Windows.Forms.Button
- b19: System.Windows.Forms.Button
+ Projekt()
- b19_Click(sender: object, e: EventArgs) : void
- b18_Click(sender: object, e: EventArgs) : void
Dispose(disposing: bool) : void
- InitializeComponent() : void

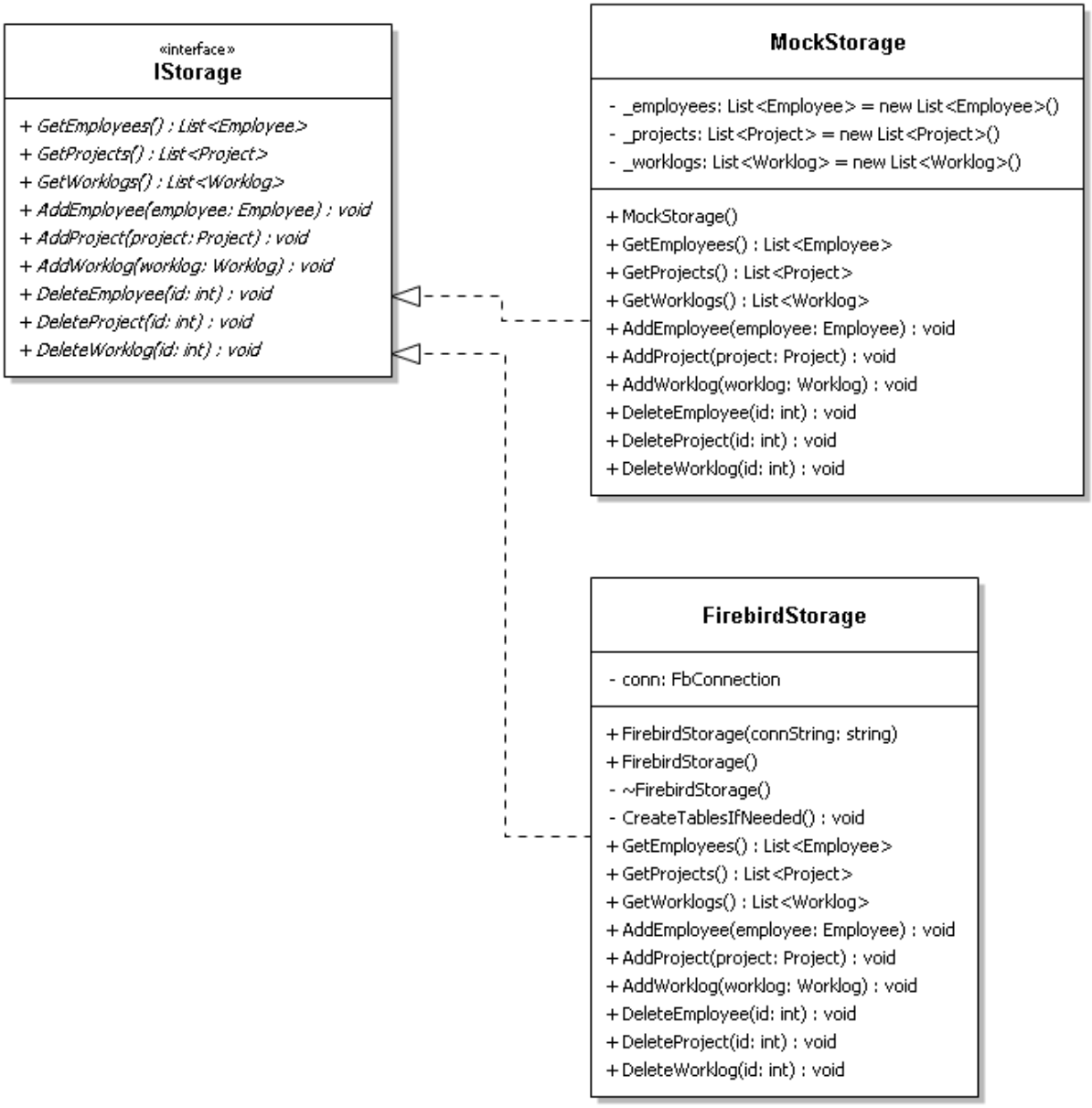
TUI



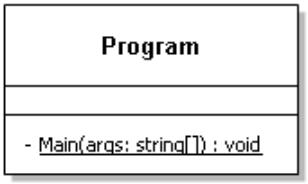
Fachkonzeptschicht

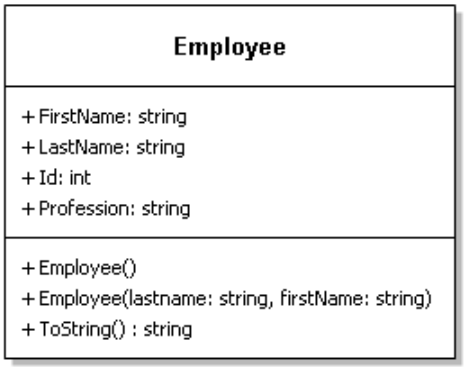


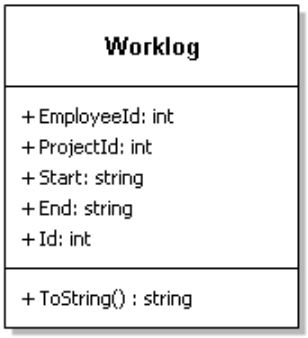
Datenhaltungsschicht

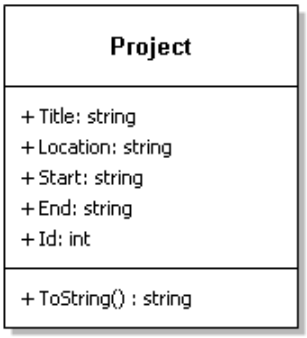


Containerklassen und Main-Klasse:

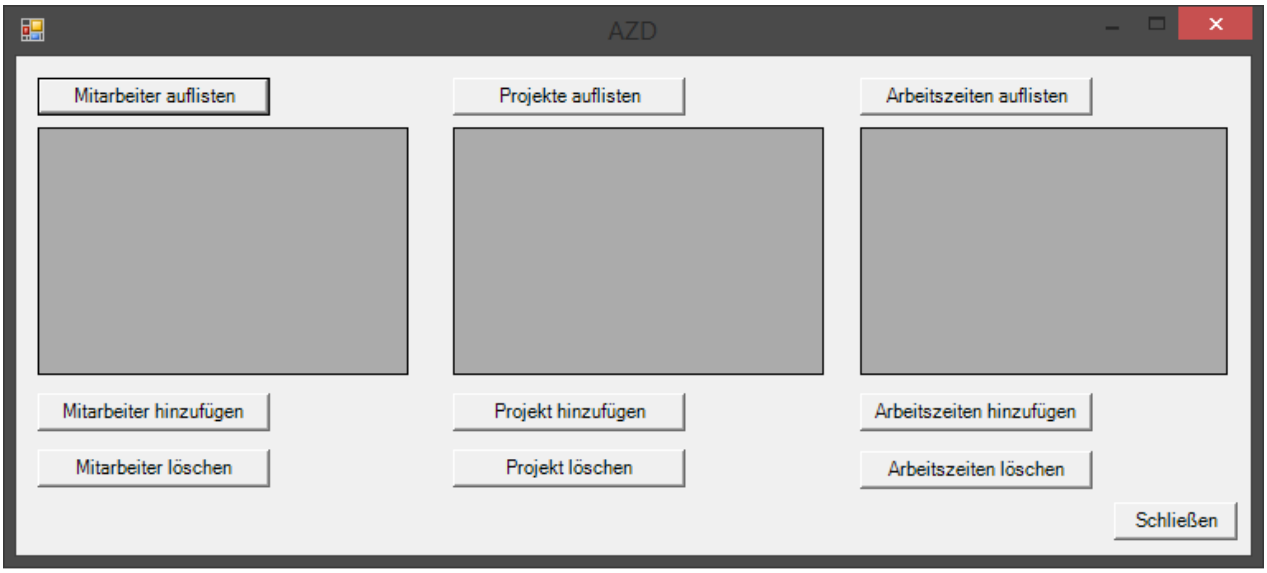


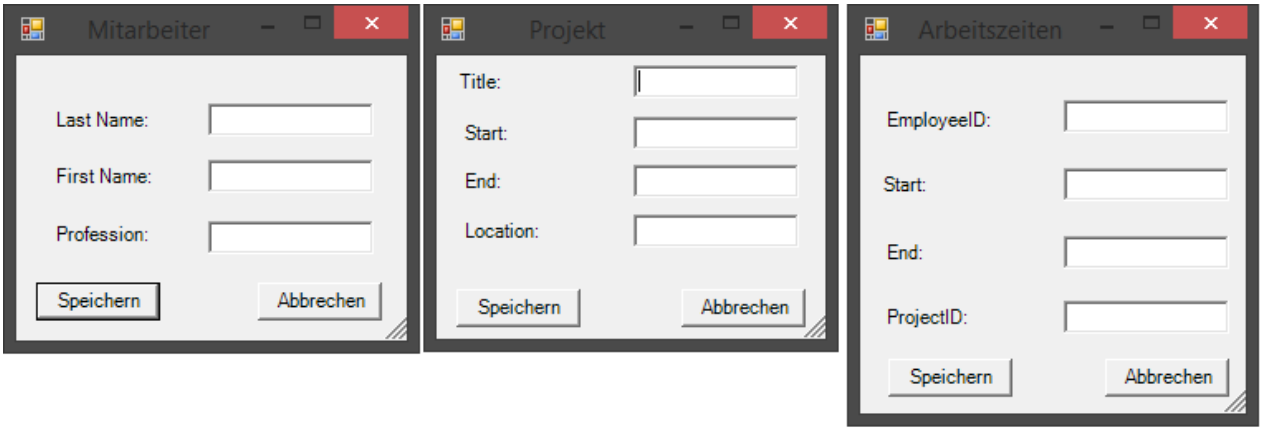


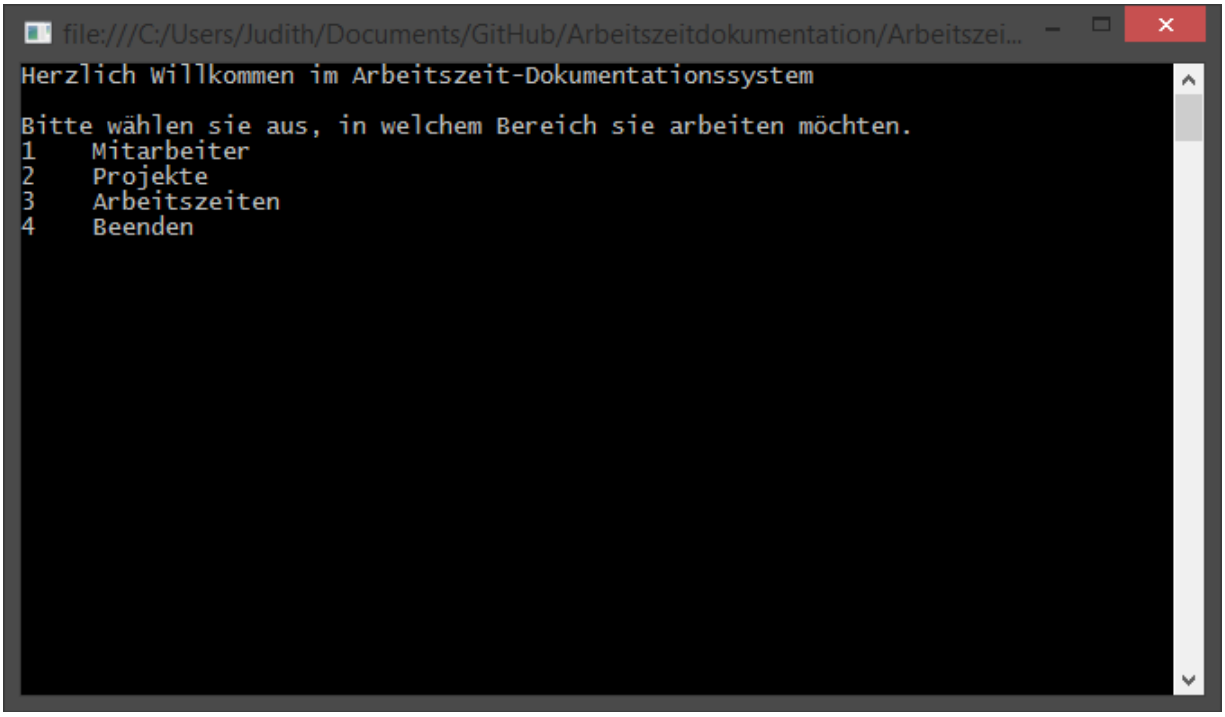




User Interfaces







Datenhaltungen

Die Datenhaltung ist auf SQL-Basis mit Firebird implementiert. Eine weitere XML-basierte Datenhaltung ist unfertig. Außerdem existiert noch die zu Beginn erstellte Klasse `MockStorage` welche das Interface `IStorage` ebenfalls vollständig implementiert, aber die Daten nicht persistent speichert, sonder nur im Arbeitsspeicher hält.

Implementation des Design-Patterns im Projekt

In diesem Projekt wurde die Sortierung von Mitarbeitern und Projekten mit dem Decorator Entwurfsmuster umgesetzt. Als abstrakte Basisklasse erbt `AbstractBusinessLogicDecorator` von dem Interface `IBusinessLogic` . Diese Klasse erwartet in ihrem Konstruktor ein Objekt welches ebenfalls `IBusinessLogic` implementiert und es delegiert alle Aufrufe standardmäßig an dieses Objekt. Von dieser Basisklasse erbt nun wiederum der konkrete Decorator `SortedBusinessLogic` . `SortedBusinessLogic` überschreibt die Methoden `GetEmployees` und `GetProjects` . Zwar leitet es weiterhin die Aufrufe an das dekorierte Objekt weiter, aber es sortiert dessen Rückgabewerte bevor es diese ebenfalls an das aufrufende Objekt zurückgegeben werden.

Schichtenkommunikation