Hochschule für Technik, Wirtschaft und Kultur Leipzig (FH)
Fachbereich Informatik, Mathematik, Naturwissenschaften

Alexander Bohne

Theoretische Grundlagen zur Implementierung eines Projektmanagement-Systems für eine Full-Service-Media-Agentur

Bachelorarbeit

Leipzig, August 2003

Erklärung

Ich versichere wahrheitsgemäß, die Arbeit selbständig, alle benutzten Hilfsmittel genau angegeben und alles kenntlich gemacht zu haben, was aus Arbeiten anderer unverändert oder mit Änderungen entnommen wurde.

1. Einleitung

1.1 XISTEM Information Technologies GmbH

Die Full-Service-Media-Agentur „XISTEM Information Technologies GmbH“, mit Ihrem Sitz in Leipzig, wurde 2002 gegründet.
Das Unternehmen liefert das ganze Leistungsspektrum einer Full-Service-Media-Agentur - angefangen vom einfachen Internet-Auftritt, über Internet-Applikationen bis hin zu komplexen Web-Content-Management-Systemen. Ein weiterer Geschäftsbereich ist die Offline-Media-Produktion, hier ist die Agentur im Bereich CD-ROM-Produktion tätig. Außerdem bietet die „XISTEM Information Technologies GmbH“ eine Betreuung bei der Planung, der Installation und dem Betrieb von Netzwerken an.
Zu Kunden des Unternehmens gehören mittelständische Unternehmen im Raum Leipzig.

1.2 Gründe zur Implementierung eines Projektmanagement-Systems

Der Definition nach DIN ist ein Projektmanagement-System ein organisatorisch abgegrenztes Ganzes, das durch das Zusammenwirken seiner Elemente in der Lage ist, Projekte vorzubereiten und abzuwickeln.

Ein solches System beinhaltet also alles, was mit der Verwaltung von einem Projekt zu tun hat. Dabei ist ein Projekt als ein komplexer Zusammenschluss verschiedenster, in sich verschachtelter und aufeinander aufbauender, zu bearbeitender Geschäftsprozesse anzusehen.

Die Vielzahl der am Markt befindlichen Projektmanagement-Systeme bieten die unterschiedlichsten qualitativen Leistungen. Dabei haben einige solcher Systeme eine extreme Spezialisierung erfahren, so dass sie auf bestimmte Einsatzgebiete optimiert sind. Zum Beispiel sind Projektmanagement-Systeme am Markt, die nur den zeitlichen Ablauf eines Projektes verwalten, andere wiederum sind auf finanz-und betriebswirtschaftliche Aspekte, wie Angebots- oder Rechnungsstellung, optimiert. Ein weiterer Themenbereich von Projektmanagement-Systemen sind Zeiterfassungs-Systeme, im Speziellen Personal-Zeiterfassungs-Systeme oder Auftrags-Zeiterfassungs-Systeme.

Komplexe Projektmanagement-Systeme sind modular aufgebaut, d. h. für verschiedene Aufgabenbereiche sind separate Module verantwortlich. So kann ein Projektmanagement-System an spezielle Anwenderwünsche angepasst werden.

Die Gründe für die Implementierung eines eigenen Projektmanagement-Systems liegen in der Optimierung eines solchen Systems auf die Anforderungen einer Full-Service-Media-Agentur, wie die „XISTEM Information Technologies GmbH“ eine ist. Dabei soll eine Spezialisierung für multimediale Projekte, sowie Projekte im Online-Media-Bereich (zum Beispiel Web-Auftritte) als auch im Offline-Media-Bereich (zum Beispiel CD-ROM-Produktion) verfolgt werden.

Auch ist ein marktrelevanter Aspekt nicht zu vernachlässigen, da ein so erstelltes Projektmanagement-System auch an andere in diesem Marktbereich tätige Unternehmen veräußert werden kann.

1.3 Aufbau dieser Arbeit

Diese Arbeit behandelt die theoretischen Grundlagen, die zur Implementierung eines Projektmanagement-Systems, welches auf die Anforderungen einer Full-Service-Media-Agentur optimiert ist, nötig sind.

Dabei werden im ersten Teil (Punkt 2. und Punkt 3.) ganz allgemein die Anforderungen an ein solches Projektmanagement-System sowie die konzeptionellen Vorstellungen eines solchen Systems erläutert. Auch werden die gewünschten Funktionalitäten der einzelnen Module, wie zum Beispiel die Kundenverwaltung oder die Mitarbeiterverwaltung, analysiert und beschrieben.

Des Weiteren werden Aspekte zur Auswahl der Systemarchitektur genauer erläutert. Außerdem behandelt diese Arbeit auch Vor- und Nachteile der einzelnen Architektur-Typen. Im Speziellen wird hier auf den Betrieb unter verschiedenen Betriebssystem-Plattformen und unter verschiedenen Datenbank-Systemen eingegangen. Außerdem wird die zu Grunde liegende Programm-Architektur (Standalone-Application, Client-Server-Architektur, etc.) diskutiert.

Im Punkt 4. wird auf die Datenbank-Struktur, wie sie in diesem Projektmanagement-System zum Einsatz kommt, eingegangen. Auch werden hier die endgültig im Produkt enthaltenen Funktionen der einzelnen Module genauestens erläutert. Zur Veranschaulichung werden zu Demonstrationszwecken dienende Eingabe-Masken dargestellt, die die jeweiligen Funktionen genauer beschreiben.

Danach (Punkt 5.) werden, im Vergleich zu anderen am Markt befindlichen Projektmanagement-Systemen, Vor- und Nachteile, sowie Aspekte zur Performance erläutert. Diese Diskussion gibt nur einen kurzen Überblick, weitere, tiefer gehende Vergleiche bilden eine Grundlage für eine separate Arbeit.
Im Folgenden (Punkt 6.) wird ein kurzer Ausblick auf zukünftige Erweiterungen des Projektmanagement-Systems, die in späteren Releases implementiert werden könnten, geworfen.

Diese Arbeit wird nur die theoretischen Grundlagen zur Implementierung eines Projektmanagement-Systems behandeln, so dass eine spätere Implementierung anhand der hier diskutierten Aspekte erfolgen kann. Die Programmierung eines solchen Systems gehört nicht zum Umfang dieser Arbeit.

2. Anforderungen und konzeptionelle Vorstellungen für den Aufbau eines Projektmanagement-Systems

2.1 Allgemeines Konzept des Projektmanagement-Systems

Das geplante Projektmanagement-System wird alle für die Ausführung eines multimedialen Projektes wichtigen Aspekte abdecken. Diese gliedern sich in verschiedene Themenbereiche (s. Abb. 2.1.1): Mitarbeiterverwaltung, Kundenverwaltung, Projektverwaltung, Partner- oder Lieferantenverwaltung, Auftragsverwaltung und einen Bereich für die Zeiterfassung der an einem Projekt tätigen Mitarbeiter.

Abbildung 2.1.1: Themenbereiche des Projektmanagement-Systems

Diese Themenbereiche dürfen nur von befugten Benutzern aufgerufen werden. So darf zum Beispiel nur ein Projektleiter verschiedene Daten seines Projektes bearbeiten, ein Systemmanager hat Zugriff auf das gesamte System. Dabei ist zu beachten, dass der Systemmanager nicht mit einem Systemadministrator zu verwechseln ist. Der Administrator ist für den korrekten Betrieb des Projektmanagement-Systems verantwortlich. Der Systemmanager hingegen ist ein Benutzer, der vollen Zugriff auf die Funktionen und somit auf die Daten des Systems hat. Im Allgemeinen besitzt beispielsweise der Geschäftsführer eines Unternehmens, das das Projektmanagement-System einsetzt, Rechte des Systemmanagers.

Es muss also eine vorherige Anmeldung am System erfolgen, die abhängig von den Benutzerrechten nur die vom jeweiligen User zugänglichen Themenbereiche des Projektmanagement-Systems zur Verfügung stellt.

2.2 Mitarbeiterverwaltung

Um ein Management der für ein Unternehmen tätigen Arbeitnehmer zu gewährleisten, bietet die Mitarbeiterverwaltung eine Möglichkeit, neue Mitarbeiterdaten anzulegen, bestehende Mitarbeiterdaten zu bearbeiten, und bisherige Mitarbeiterdaten, bei Ausscheiden aus dem Unternehmen, zu löschen. Abbildung 2.2.1 zeigt die Integration der Mitarbeiterverwaltung in das Projektmanagement-System.

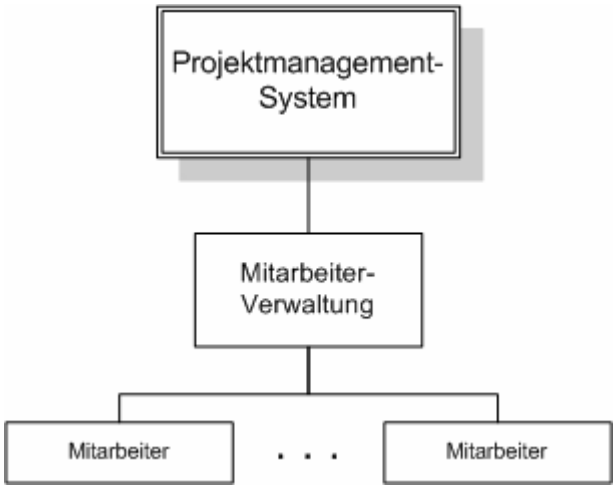


Abbildung 2.2.1: Integration der Mitarbeiterverwaltung in das Projektmanagement-System

Es werden alle für das Unternehmen wichtigen Eckdaten eines Mitarbeiters gespeichert. Diese sind hauptsächlich persönlicher Natur, wie zum Beispiel Namen, Adressen, Telefonnummern. Aber auch finanztechnische Daten werden hier verwaltet, solche sind beispielsweise Bankverbindungsdaten oder auch Daten zur Krankenversicherung.

Des Weiteren ist es wünschenswert, mit dem arbeitgebenden Unternehmen vertraglich gebundene Daten, wie vereinbarte Arbeitszeiten oder Ähnliches, zu bearbeiten.
Aus Sicherheits- und datenschutzrechtlichen Gründen darf dieses Mitarbeitermanagement nur von autorisierten Benutzern vorgenommen werden.
Weiterhin bietet die Mitarbeiterverwaltung eine Definition der einzelnen Benutzerrechte (Systemmanager, Projektleiter, Hauptbenutzer, etc.) sowie die Speicherung der Zugangsdaten der Mitarbeiter zum Projektmanagement-System.
Mitarbeiterdaten darf nur ein Benutzer mit Rechten des Systemmanagers bearbeiten.

2.3 Kundenverwaltung

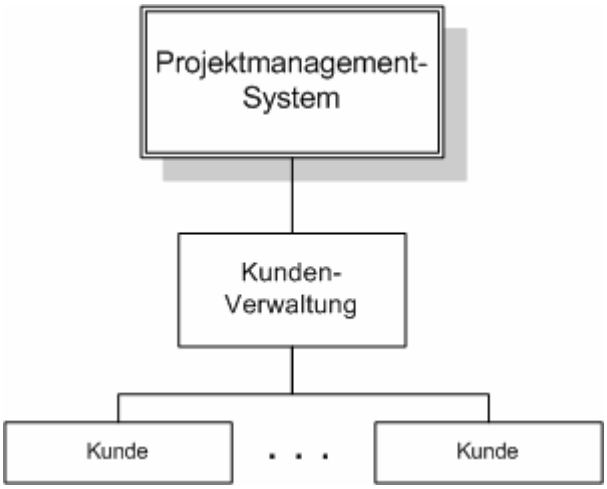


Abbildung 2.3.1 Integration der Kundenverwaltung in das Projektmanagement-System

Abbildung 2.3.1 stellt die Einbindung des Themenbereiches der Kundenverwaltung in das Projektmanagement-System dar. Mit dieser Kundenverwaltung ist ein Management der Kunden/Auftraggeber des Unternehmens möglich.

Der Themenbereich bietet auch hier die Möglichkeit, neue Kundendaten anzulegen, vorhandene Kundendaten zu bearbeiten und Daten von Kunden zu löschen.

Diese Kundendaten sind Adressdaten, aber auch Daten finanztechnischer Art, wie ein vereinbartes Zahlungsziel oder auch ein etwaiger Rabatt.

Diese Daten dürfen wiederum nur von einem Systemmanager bearbeitet werden.

2.4 Projektverwaltung

Im Themenbereich der Projektverwaltung kann ein Projektleiter oder ein Systemmanager neue Projekte anlegen und Projektdaten bearbeiten. Abbildung 2.4.1 zeigt ein Schema, welches die Zusammenarbeit der Projektverwaltung mit der Kundenverwaltung darstellt.

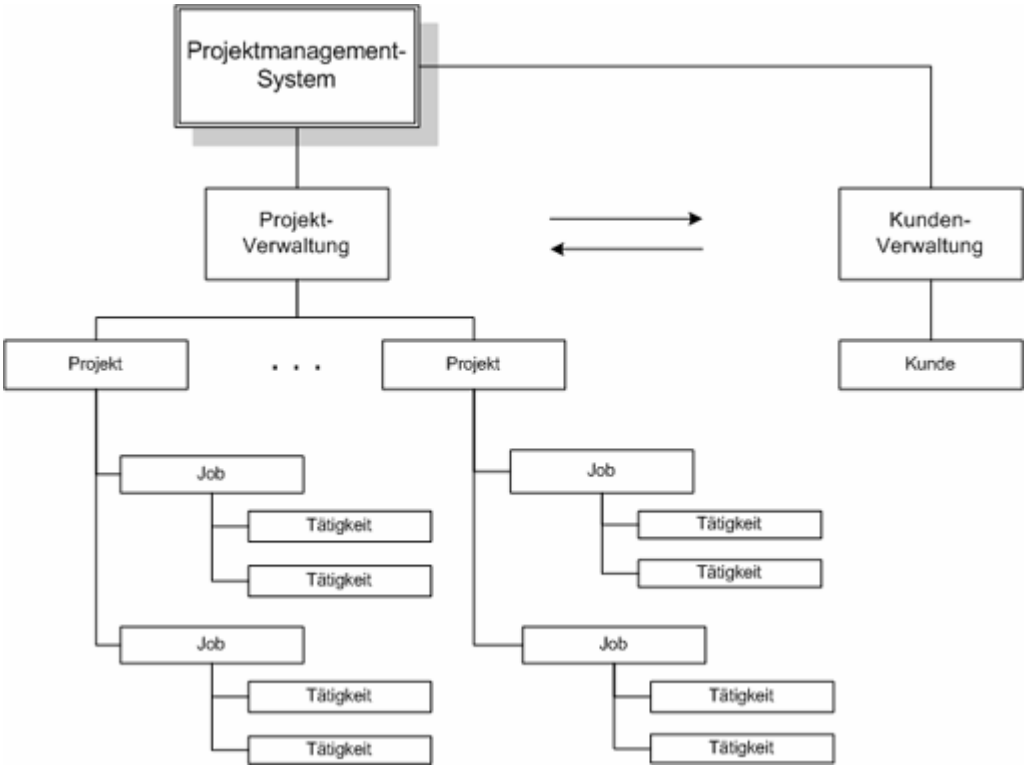


Abbildung 2.4.1: Integration der Projektverwaltung in das Projektmanagement-System

Beim Neuanlegen eines Projektes werden zunächst die Daten des Kunden/Auftraggeber, für den dieses Projekt realisiert wird, eingegeben.

Bei Nichtvorhandensein eines projektbezogenen Kunden wird zur Kundenverwaltung verwiesen, so dass dort der neue Kunde/Auftraggeber anzulegen ist.

Des Weiteren bietet die Projektverwaltung die Möglichkeit, alle für die Ausführung eines Projektes erforderlichen Einzel-Jobs, die später in eine Rechnung einfließen werden, anzulegen. Eine weitere feinere Separierung der Jobs wird danach durch Aufteilungen in einzelne Tätigkeiten realisiert. Zu den Daten dieser Tätigkeiten gehören wiederum auch deren Beginn und Ende, so dass die Möglichkeit besteht, einen zeitlichen Überblick über den Ablauf eines Projektes zu erhalten. Außerdem kann die Ausführung von separaten Jobs externe Unternehmen abgetreten werden.

Weitere Projektdaten sind beispielsweise der Name eines Projektleiters und eine eindeutige interne Projektnummer.

Diese Daten darf wiederum nur ein Systemmanager oder ein vom Systemmanager bestimmter Projektleiter bearbeiten.

2.5 Auftragsverwaltung

In der Auftragsverwaltung werden projektbezogene Auftragsdaten neu angelegt und bisherige Daten bearbeitet. Dieser Themenbereich arbeitet mit der Projektverwaltung und somit auch indirekt mit der Kundenverwaltung zusammen, wie aus Abbildung 2.5.1 hervorgeht.

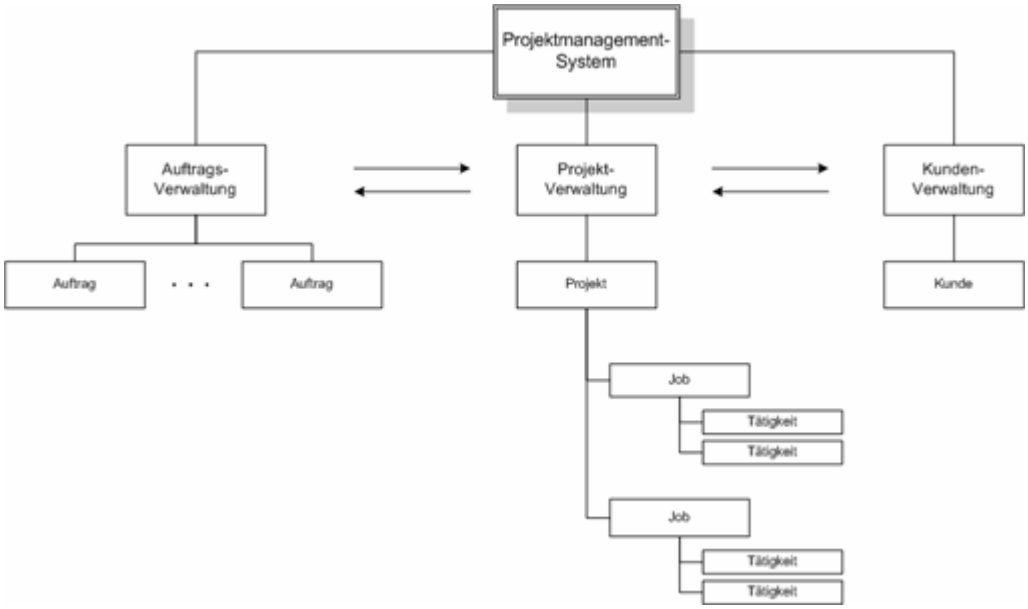


Abbildung 2.5.1: Integration der Auftragsverwaltung in das Projektmanagement-System

Wichtige Auftragsdaten zu einem Projekt sind ein Termin für eine Angebotsabgabe an einen potentiellen Auftraggeber – die Angebotserstellung selbst wird auch ein Feature der Auftragsverwaltung sein, da in der Projektverwaltung die Einzel-Jobs und Einzel-Tätigkeiten eines Projektes angelegt wurden, die jetzt in einem Angebot aufgelistet werden.

Hierzu ist dann nur noch eine preisliche Kalkulation der separaten Jobs bzw. Tätigkeiten notwendig.

Weitere Daten zur Auftragsverwaltung sind Termine zur Auftragserteilung, zur Rechnungslegung und zum Projektabschluss.

Die Auftragsverwaltung arbeitet eng mit der Projektverwaltung zusammen, so dass auch hier nur ein Projektleiter oder ein Systemmanager diese Daten bearbeiten darf.

2.6 Partner-/Lieferantenverwaltung

Manche Tätigkeiten werden von einem externen Partner oder Lieferanten ausgeführt. Beispiele dafür wären zum Beispiel das Drucken von Plakaten oder Flyern. Daher besteht in der Partner- oder Lieferantenverwaltung die Möglichkeit, diese Daten von externen Partnern bzw. Lieferanten, zu bearbeiten. Abbildung 2.6.1 stellt nun die Integration der Partnerverwaltung in das Projektmanagement-System dar.

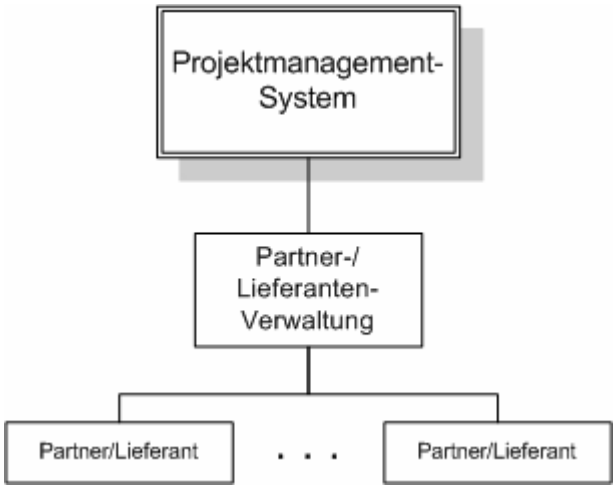


Abbildung 2.6.1: Integration der Partner-/Lieferantenverwaltung in das Projektmanagement-System

Solche Daten sind Adressdaten, wie Name, Anschrift, Telefonnummern, etc. Auch wird ein Ansprechpartner verzeichnet.
Eine eindeutige Identifikationsnummer spezifiziert einen vorhandenen Partner/Lieferanten.
Diese Daten dürfen nur von einem Projektleiter oder von einem Systemmanager editiert werden.

2.7 Zeiterfassung

In dem Bereich der Zeiterfassung bearbeitet ein am System angemeldeter Mitarbeiter seine, an einem bestimmten Projekt erbrachte, Arbeitszeit. Abbildung 2.7.1 zeigt die Einbindung der Zeiterfassung in das Projektmanagement-System.

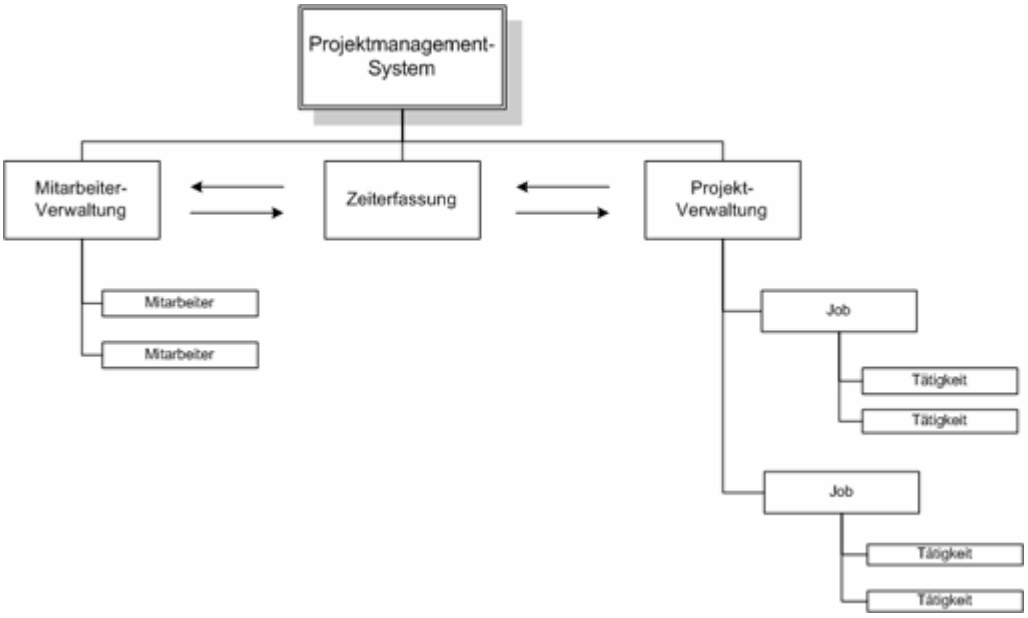


Abbildung 2.7.1: Integration der Zeiterfassung in das Projektmanagement-System

Die Vorgehensweise des Eintragens wird diese sein, dass ein Mitarbeiter zu einem bestimmten Datum die geleistete Arbeitszeit an einem Job oder an einer Tätigkeit eines Projektes einträgt. Durch eine von einem Systemmanager in der Mitarbeiterverwaltung eingetragene vertraglich gebundene Mindestarbeitszeit erhält der Mitarbeiter eine Übersicht über die aktuell erbrachte Arbeitszeit bzw. notwendige Mehrarbeitszeit. Ein Projektleiter erfährt des

Weiteren die für einen Job erbrachten Stundenanzahl, diese werden für weitere geschäftliche Kalkulationen, beispielsweise zur Rechnungserstellung, benutzt.

Zeiterfassungsdaten dürfen nur in einem begrenzten Zeitraum editierbar sein, so muss zum Beispiel eine in der Zukunft liegende Eintragung unmöglich sein, wie auch eine schon eingetragene Zeiterfassung nur am gleichen Tag verändert werden darf.

3. Programmtechnische Grundlagen

3.1 Aspekte zur Auswahl der Systemarchitektur

Da in dieser Arbeit nur die theoretischen Grundlagen für die Implementierung eines Projektmanagement-Systems erläutert werden, können auch hier nur grundlegende Hinweise, sowie Vor- bzw. Nachteile der verschiedenen Systemarchitekturen gegeben werden.

Die zwei Hauptkriterien sind dabei einmal, das Projektmanagement-System als eigenständigen Applikationsserver (s. Abb. 3.1.1) zu erstellen und zu betreiben, zum Anderen besteht die Möglichkeit, das Projektmanagement-System als Applikation eines zu Grunde liegenden Servers zu betreiben. In jedem der zwei Anwendungsfälle stellt eine Server-Applikation einem Client, über eine ansprechende grafische Benutzeroberfläche (Frontend), die programmtechnischen Funktionalitäten zur Verfügung.

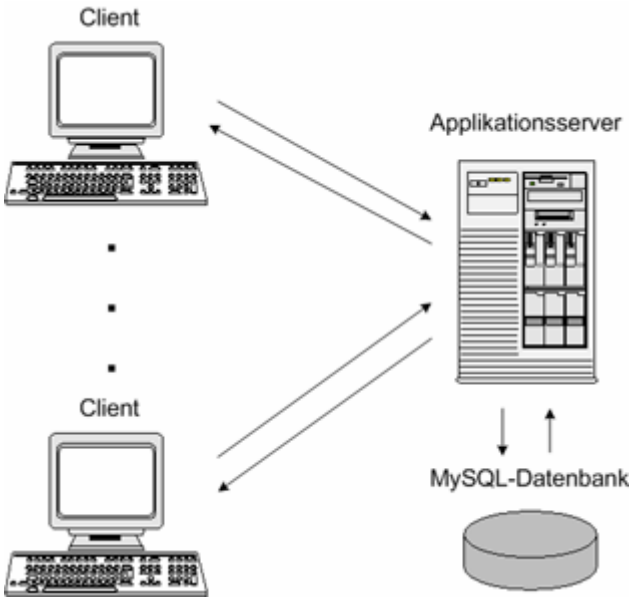


Abbildung 3.1.1: Arbeitsweise des Projektmanagement-Systems als eigenständiger Applikationsserver

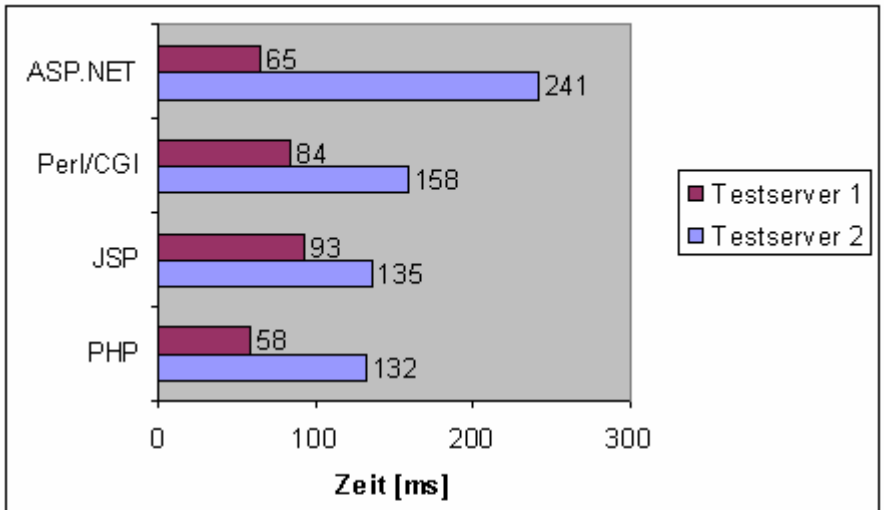
Abbildung 3.1.1 stellt die Arbeitsweise des Projektmanagement-Systems als eigenständigen Applikationsserver dar. Vorteile für diesen Betrieb sind die kompakte und einfach zu administrierende Arbeitsumgebung. Diese kann beispielsweise als Java-Applikation realisiert werden. Dabei stellt ein eigener HTTP-Server, der auch als Java-Applikation implementiert wird, die Kommunikation zwischen Client und Projektmanagement-System zur Verfügung. Der Zugriff der Clients auf den Server, und damit auf die Funktionalitäten des Projektmanagement-Systems, erfolgt dabei über einen Webbrowser oder auch über eine native Java-Applikation.

Ein separates Datenbank-System stellt die Funktionen zur Speicherung aller zum Betrieb des Projektmanagement-Systems nötigen Daten zur Verfügung.

Eine andere Funktionsweise ist ein Betrieb als Applikation unter einem vorhandenen Server-System.
Beispielsweise kann das Projektmanagement-System als PHP-Projekt unter einem generischen HTTP-Server betrieben werden. Dabei wird ein HTTP-Server, auf dem die PHP-Erweiterung installiert ist, die Funktionen des Projektmanagement-Systems zur Verfügung stellen. Die Kommunikation zwischen Client und Server realisiert dabei ein Webbrowser.

Welche Programmierart (PHP, ASP/ASP.NET, Perl, JSP, Java-Servlet) bei der Implementierung des Projektmanagement-Systems zum Einsatz kommt, hängt hauptsächlich vom eingesetzten Webserver und dessen Konfiguration ab. Für das Ausführen der jeweiligen Programmdateien auf dem HTTP-Server muss die dazugehörige Server-Erweiterung installiert werden. Beispielsweise ist zum Ausführen von PHP-Dateien unter dem Microsoft Internet Information Server die PHP-Erweiterung zu installieren. Active Server Pages (ASP) werden jedoch standardmäßig schon unterstützt, für ASP.NET-Unterstützung muss das .NET Framework installiert werden. Anders der Apache-HTTP-Server: dieser hat die PHP-Erweiterung schon integriert, für die ASP-Unterstützung muss hier eine Erweiterung installiert werden. Für die Perl-, JSP- und Servlet-Unterstützung muss in jedem Fall ein Plug-In installiert werden.

Abbildung 3.1.2 stellt einen Performance-Vergleich verschiedener Programmierarten unter 2 verschiedenen Testservern dar. Dabei erfolgt ein Auslesen von 100 Datensätzen aus einer MySQL-Datenbank und deren Ausgabe im Browser des Clients. Mit Hilfe von Skripten, die in ASP.NET, PHP, Perl und Java als JSP programmiert sind, wird die durchschnittlich benötigte Zeit, die zum Auslesen der Datensätze nötig ist, gemessen und im Browser angezeigt.



Testserver 1:	Testserver 2:
Pentium IV - 2.0 GHz	Pentium II - 350 MHz
256 MB DDR-RAM	160 MB SDR-RAM
MS Windows XP Professional	MS Windows 2000 Server
MS Internet Information Server 5.1	MS Internet Information Server 5.0
MS .NET Framework 1.0	MS .NET Framework 1.0
PHP 4.3.1	PHP 4.3.1
ActiveState Perl 3.8.0	ActiveState Perl 3.8.0
ServletExec 4.2	ServletExec 4.2
MySQL 3.23.49	MySQL 3.23.49

Abbildung 3.1.2: Performance-Vergleich verschiedener Programmierarten unter verschiedenen Plattformen

Wie aus der Abbildung 3.1.2 ersichtlich, ist beim Auslesen von 100 Datensätzen aus einer MySQL-Datenbank das PHP-Skript unter beiden Testservern am schnellsten.

Beachtlich ist auch der Geschwindigkeitszuwachs des ASP.NET-Skriptes unter dem schnelleren Testserver 1. Da dieses unter dem Testserver 2 nur durchschnittlich 241ms für das Auslesen der 100 Datensätze benötigt und unter dem Testserver 1 eine Zugriffszeit von 65ms hat, ist davon auszugehen, dass das .NET Framework besonders von der Hardware des Servers abhängig ist.
Die Ausführungszeiten des Perl- und des JSP-Skriptes liegen im Mittelfeld.
Der Quellcode der 4 verschiedenen Testskripte befindet sich im Anhang.

Für den Betrieb in einer Full-Service-Media-Agentur empfiehlt sich daher der Betrieb als Server-Applikation, etwa als PHP-Projekt unter einem existierenden Web-Server und einer MySQL-Datenbank, wie in Abbildung 3.1.3 dargestellt ist.

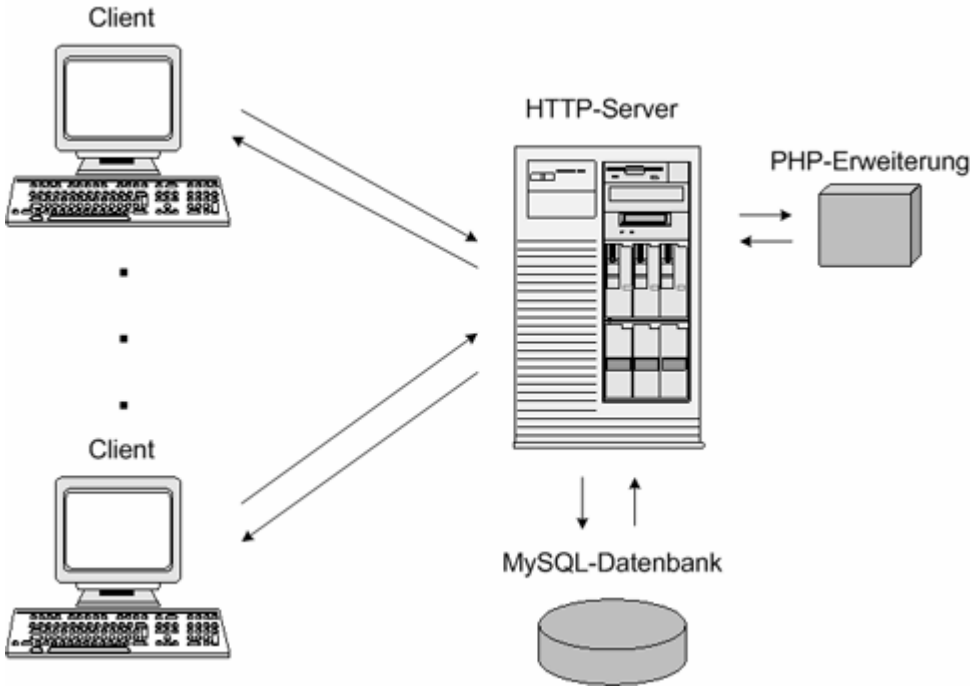


Abbildung 3.1.3: Arbeitsweise des Projektmanagement-Systems als PHP-Projekt unter einem HTTP-Server mit einer MySQL-Datenbank

Ein vorhandener HTTP-Server hat gegenüber einem integrierten Applikationsserver den Vorteil, dass dieser schon auf parallele clientseitige Anfragen optimiert ist. Bei einem Applikationsserver müsste dies erst implementiert werden. Auch bietet ein existierender HTTP-Server den Vorteil einer serverseitigen HTTP-Komprimierung an, die eine Netzauslastung in Grenzen hält.
Des Weiteren hat dieses Funktionsprinzip den Vorteil, dass immer nur die gerade in Benutzung befindlichen Programmteile den Server beanspruchen, was wiederum die Performance verbessert. Beim Betrieb als eigenständiger Applikationsserver lastet das gesamte System den Host-Computer aus, da dieses nur als Ganzes so funktionsfähig ist.

Auch sicherheitstechnisch hat die Konstellation einer Server-Applikation Vorteile, da bei Auftauchen von Fehlern oder Sicherheitslöchern in den HTTP-Servern diese nach spätestens 4-5 Tagen durch vom Hersteller bereitgestellte Patches behoben werden. Beim Betrieb des Projektmanagement-Systems als eigenständigen Applikationsserver muss zur Fehlerkorrektur bzw. zum Sicherheitsupdate immer das gesamte System neu erstellt werden, d.h. der entsprechenden Code-Abschnitt muss geändert werden, die komplette Applikation neu erstellt/kompiliert und neu aufgesetzt werden.

Für den Betrieb in einer Full-Service-Media-Agentur wird von maximal 50 Benutzern ausgegangen, die mit dem Projektmanagement-System arbeiten. Daher ist ein Datenbank-System mit dem Funktionsumfang des MySQL-Datenbank-Systems völlig ausreichend. Abhängig von der Hardware des Servers, besonders von der Größe des Hauptspeichers, wird ein solches System erst bei einer Bearbeitung von ca. 100.000 Datensätzen spürbar langsamer.
Andere Datenbank-Systeme vom Leistungsumfanges eines Oracle- oder DB2-Datenbank-Systems sind für die Funktionen eines Projektmanagement-Systems zwar geeignet aber völlig überdimensioniert, da bei der generierten Serverlast keine zeitkritischen Zustände entstehen und so beispielsweise weder verteilte Datenbanken oder ein Transaktions-Management nötig sind.

Ein weiterer, nicht zu vernachlässigender Aspekt ist auch der fast kostenlose Betrieb in einer Betriebsart als Server-Applikation. HTTP-Server gibt es als Open-Source, beispielsweise der Apache-HTTP-Server, der eine große Nutzerschar besitzt. Die PHP-Erweiterung wird ebenfalls als Open-Source vertrieben. Einzig für das Datenbank-System fallen Kosten an. Zum Beispiel sind für eine Lizenz des MySQL-Datenbank-Systems 275,00 €[1] inkl. MWSt. an den Hersteller zu entrichten.

[1] Stand Oktober 2003

4. Programmtechnische Umsetzung

4.1 Datenbank-Struktur

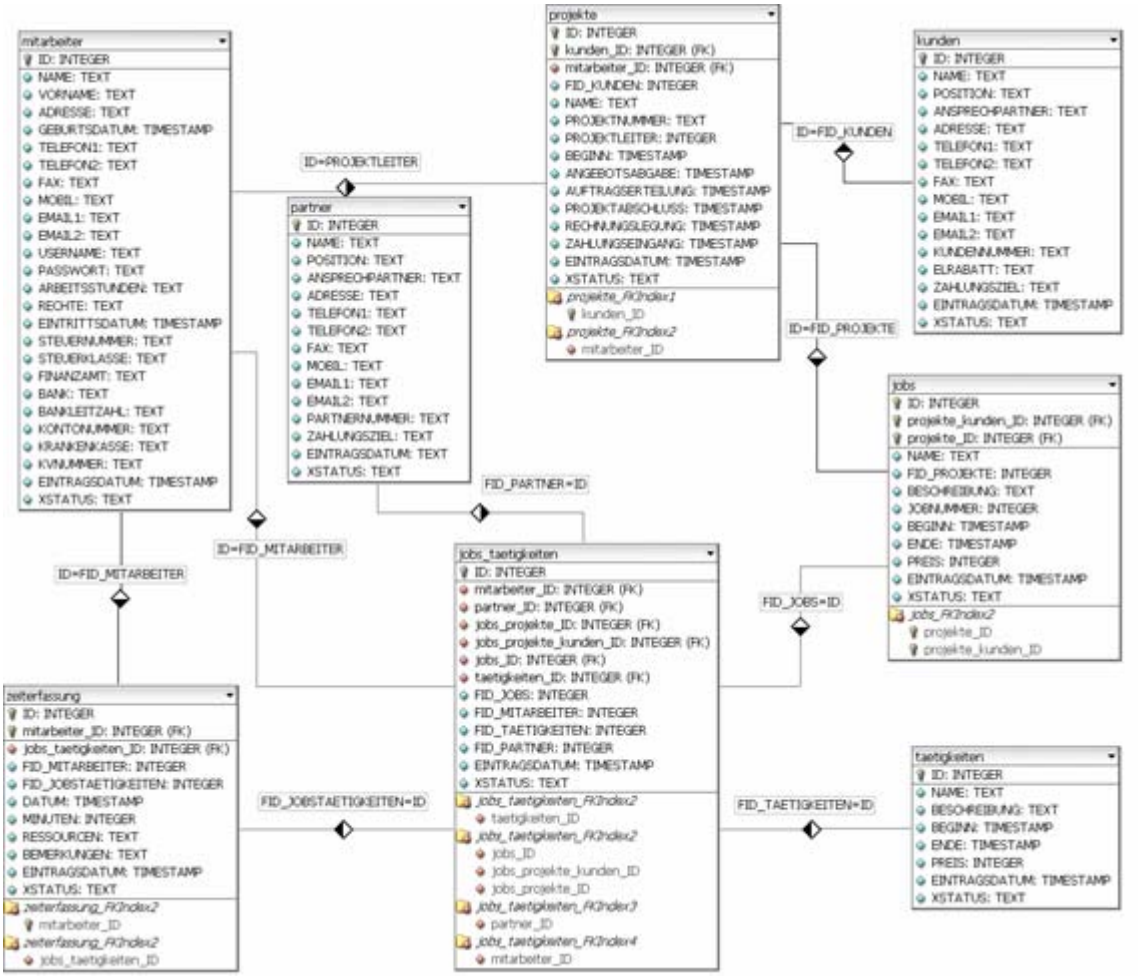


Abbildung 4.1.1: Datenbank-Schema des Projektmanagement-Systems

Um die gewünschten Funktionalitäten des zu implementierenden Projektmanagement-Systems zu gewährleisten, ist eine funktional-korrekte Datenbank-Struktur nötig. Ein solches Datenbank-Schema, welches Abbildung 4.1.1 zeigt, ist der Kern des gesamten Projektmanagement-Systems.

Für jeden zu managenden Bereich (Mitarbeiterverwaltung, Projektverwaltung, Kundenverwaltung, etc.) steht eine Datenbanktabelle (Entität) zur Verfügung, die die jeweiligen Daten speichert. Dabei sind die einzelnen Entitäten so miteinander verknüpft, dass ein logischer Zusammenhang untereinander besteht - bestimmte Tabellen sind dabei, bedingt durch eine Normalisierung des Schemas, über so genannte Kreuztabellen verbunden.
Über verschiedene Frontends wird beim Betrieb des Projektmanagement-Systems auf die einzelnen Datenbanktabellen zugegriffen, um eine Bearbeitung derer zu gewährleisten. Beispielsweise werden beim Neuanlegen eines Mitarbeiters in der Mitarbeiterverwaltung die in den Frontends eingegebenen Daten in die Datenbanktabelle „mitarbeiter“ eingefügt. Oder falls Kundendaten geändert werden sollen, zum Beispiel eine Änderung der Faxnummer, wird in dem zum Kunden gehörenden Datensatz der Tabelle „kunden“ die Faxnummer geändert.

Genauere Hinweise befinden sich in den zur jeweiligen Datenbanktabelle gehörenden Texten.

Im Anhang befindet sich, der besseren Übersicht wegen, eine vergrößerte Darstellung des Datenbank-Schemas aus Abbildung 4.1.1.

Abbildung 4.1.2 zeigt die Tabelle „mitarbeiter“. Sie bildet den Hintergrund zur Speicherung und Bearbeitung aller Mitarbeiterdaten.
Diese Daten sind alle persönlicher (NAME, VORNAME, ADRESSE, TELEFON1, etc.) wie auch finanztechnischer (BANK, BLZ, FINANZAMT, KRANKENVERSICHERUNG, etc.) Natur. Des Weiteren sind hier auch die Zugangsdaten (USERNAME, PASSWORT) zum Projektmanagement-System hinterlegt. Das Passwort ist als SHA1-Hash verschlüsselt gespeichert.

Tabelle „mitarbeiter“		
Name	Typ	Beschreibung
ID	integer	eindeutige Identifikationsnummer des Datensatzes
NAME	text	Familienname des Mitarbeiters
VORNAME	text	Vorname des Mitarbeiters
ADRESSE	text	Adresse des Mitarbeiters (Strasse, Hausnummer, Postleitzahl, Wohnort)
GEBURTSDATUM	timestamp	Geburtsdatum des Mitarbeiters
TELEFON1	text	erste Telefonnummer des Mitarbeiters
TELEFON2	text	zweite Telefonnummer des Mitarbeiters (optional)
FAX	text	Faxnummer des Mitarbeiters (optional)
MOBIL	text	Mobiltelefonnummer des Mitarbeiters (optional)
EMAIL1	text	erste E-Mail-Adresse des Mitarbeiters
EMAIL2	text	zweite E-Mail-Adresse des Mitarbeiters (optional)
USERNAME	text	Benutzername des Mitarbeiters, dient zur Anmeldung am System
PASSWORT	text	Passwort des Mitarbeiters, dient zur Anmeldung am System, als SHA1-Hash gespeichert

Abbildung 4.2.1: Datenbanktabelle „mitarbeiter“ (Forts. auf S. 31)

ARBEITSSTUNDEN	text	vereinbarte Arbeitsstundenzahl des Mitarbeiters
RECHTE	text	Rechte des Mitarbeiters (Normal, Projektleiter, Systemmanager)
EINTRITTSDATUM	timestamp	Eintrittsdatum des Mitarbeiters ins Unternehmen
STEUERNUMMER	text	Steuernummer des Mitarbeiters
STEUERKLASSE	text	Steuerklasse des Mitarbeiters
FINANZAMT	text	für den Mitarbeiter zuständiges Finanzamt
BANK	text	Name des kontoführenden Institutes des Mitarbeiters
BANKLEITZAHL	text	Bankleitzahl des kontoführenden Institutes des Mitarbeiters
KONTONUMMER	text	Kontonummer des Mitarbeiters
KRANKENKASSE	text	Name des krankenversichernden Institutes des Mitarbeiters
KVNUMMER	text	Krankenversicherungsnummer des Mitarbeiters
EINTRAGSDATUM	timestamp	Datum des Eintrages des Datensatzes in die Datenbank
XSTATUS	text	Status des Eintrages des Datensatzes (aktiv, gelöscht)

Abbildung 4.1.2: Datenbanktabelle „mitarbeiter“ (Forts. von S. 30)

Die Rechteverwaltung wird hierarchisch strukturiert.

„Normal“ steht für einen generischen Mitarbeiter, der nur Zugang zum Zeiterfassungs-System hat.

Die nächst höhere Stufe ist „Projektleiter“, der zudem Zugang zur Projektverwaltung, Kundenverwaltung und Partner-/Lieferantenverwaltung hat.

Die oberste Stufe ist der „Systemmanager“, welcher zu den vorigen Rechten auch alle persönlichen Daten über die Mitarbeiter mit Rechten unterer - und auch gleichgestellter - Stufen verwalten darf.

In der Tabelle „kunden“ (Abbildung 4.1.3) werden alle für das Projektmanagement-System relevanten Daten der Kunden/Auftraggeber gespeichert.
Eine eindeutige Kundennummer spezifiziert einen einzelnen Kunden. Auch wird hier mit jedem Auftraggeber ein Zahlungsziel vereinbart, nach welchem offene Entgelte zu begleichen sind. Dabei ist das Zahlungsziel als Anzahl Tage anzusehen. Des Weiteren wird ein möglicher Rabatt, beispielsweise für Eigenleistungen, gespeichert. Kontaktdaten, wie beispielsweise NAME, ADRESSE, EMAIL1, etc., komplettieren die Speicherung von Kundendaten.

Tabelle „kunden“		
Name	Typ	Beschreibung
ID	integer	eindeutige Identifikationsnummer des Datensatzes
NAME	text	Name des Kunden (Firma)
POSITION	text	Position des Ansprechpartners
ANSPRECHPARTNER	text	Name des Ansprechpartners)
ADRESSE	text	Adresse des Kunden
TELEFON1	text	erste Telefonnummer
TELEFON2	text	zweite Telefonnummer (optional)
FAX	text	Faxnummer des Kunden
MOBIL	text	Mobiltelefonnummer
EMAIL1	text	erste E-Mail-Adresse
EMAIL2	text	zweite E-Mail-Adresse (optional)
KUNDENNUMMER	text	eindeutige interne Kundennummer
ELRABATT	text	möglicher fester vereinbarter Rabatt des Kunden
ZAHLUNGSZIEL	text	vereinbartes Zahlungsziel
EINTRAGSDATUM	timestamp	Datum des Eintrages des Datensatzes in die Datenbank
XSTATUS	text	Status des Datensatzes (aktiv, gelöscht)

Abbildung 4.1.3: Datenbanktabelle „kunden“

Die Datenbanktabelle „partner“ (Abbildung 4.1.4) gewährleistet die Bearbeitung der Daten von Partnern bzw. Lieferanten.
Ähnlich wie auch die Tabelle „kunden“ beinhaltet sie alle für das Projektmanagement-System relevanten Daten. Auch ist hier wieder ein Zahlungsziel, in dessen Zeitraum offene Entgelte beglichen werden sollen, gespeichert. Nur kann in diesem Falle ein Partner auch ein potentieller Auftragnehmer sein. Dann ist das Zahlungsziel, in umgekehrter Richtung, die Frist zur Begleichung eigener Rechnungen. Des Weiteren sind Daten wie NAME, ADRESSE, TELEFON1, EMAIL1, etc. für eine Kontaktverwaltung von Partnern unerlässlich.

Tabelle „partner“		
Name	Typ	Beschreibung
ID	integer	eindeutige Identifikationsnummer des Datensatzes
NAME	text	Name des Partners/Lieferanten (Firma)
POSITION	text	Position des Ansprechpartners in Partnerunternehmen (Geschäftsführer, Teamleiter, Mitarbeiter)
ANSPRECHPARTNER	text	Name des Ansprechpartner im Partnerunternehmen
ADRESSE	text	Adresse des Partners/Lieferanten (Strasse, Hausnummer, Postleitzahl, Ort)
TELEFON1	text	erste Telefonnummer des Partners/Lieferanten
TELEFON2	text	zweite Telefonnummer des Partners/Lieferanten (optional)
FAX	text	Faxnummer des Partners/Lieferanten
MOBIL	text	Mobiltelefonnummer des Partners/Lieferanten

Abbildung 4.1.4: Datenbanktabelle „partner“ (Forts. auf S. 34)

EMAIL1	text	erste E-Mail-Adresse des Partners/Lieferanten
EMAIL2	text	zweite E-Mail-Adresse des Partners/Lieferanten
PARTNERNUMMER	text	eindeutige interne Partnernummer
ZAHLUNGSZIEL	text	vereinbartes Zahlungsziel (Anzahl Tage)
EINTRAGSDATUM	timestamp	Datum des Eintrages des Datensatzes
XSTATUS	text	Status des Datensatzes (aktiv, gelöscht)

Abbildung 4.1.4: Datenbanktabelle „partner“ (Forts. von S. 33)

Projektdaten werden in der Datenbanktabelle „projekte“ (Abbildung 4.1.5) gespeichert. Eine eindeutige Projektnummer spezifiziert ein einzelnes Projekt, und ein Systemmanager bestimmt einen Projektleiter, der aus der Datenbanktabelle „mitarbeiter“ über den Fremdschlüssel „PROJEKTLEITER“ referenziert wird. Da ein Projekt in den meisten Fällen an einen Kunden gebunden ist (außer interne Projekte), wird auch die Datenbanktabelle „kunden“ über einen Fremdschlüssel „FID_KUNDEN“ referenziert.
Weitere zu speichernde Daten sind das Datum des Projektbeginnes, ein Datum der Angebotsabgabe, Daten der Auftragserteilung, Rechnungslegung und das Datum des Zahlungseinganges. Diese Daten sollen einen zeitlichen Überblick über das gesamte Projekt gewährleisten.

Tabelle „projekte“		
Name	Typ	Beschreibung
ID	integer	eindeutige Identifikationsnummer des Datensatzes
FID_KUNDEN	integer	Fremd-ID des Kunden, Fremdschlüssel
NAME	text	Name des Projektes
PROJEKTNUMMER	text	eindeutige interne Projektnummer
PROJEKTLEITER	integer	Fremd-ID des Mitarbeiters, der als Projektleiter eingeteilt wurde
BEGINN	text	Datum des Beginns des Projektes
ANGEBOTSABGABE	timestamp	Datum der Abgabe eines Angebotes
AUFTRAGSERTEILUNG	timestamp	Datum der Erteilung des Auftrages
PROJEKTABSCHLUSS	timestamp	Datum des Projektabschlusses
RECHNUNGSLEGUNG	timestamp	Datum der Rechnungslegung

Abbildung 4.1.5: Datenbanktabelle „projekte“ (Forts. auf S. 36)

ZAHLUNGSEINGANG	timestamp	Datum des Einganges der Entgeltung
EINTRAGSDATUM	timestamp	Datum des Eintrages des Datensatzes in die Datenbank
XSTATUS	text	Status des Datensatzes (aktiv, gelöscht)

Abbildung 4.1.5: Datenbanktabelle „projekte“ (Forts. von S. 35)

Einzelne auszuführende Arbeiten an einem Projekt werden von einem Projektleiter in so genannte Jobs aufgeteilt. Abbildung 4.1.6 zeigt nun diese Datenbanktabelle. Dabei referenziert ein Fremdschlüssel „FID_PROJEKTE“ die Datenbanktabelle „projekte“, der eine Definition der zu einem Projekt gehörenden Jobs realisiert. Solche Jobs werden dann weiterhin in verschiedene Tätigkeiten separiert. Zu beachten ist, dass hier der XSTATUS nicht „aktiv“ oder „inaktiv“ ist, sondern „fertig“ oder „unfertig“ ist. Dabei zeigt dieser Status den Bearbeitungsstand des jeweiligen Jobs an.

Tabelle „jobs“		
Name	Typ	Beschreibung
ID	integer	eindeutige Identifikationsnummer des Datensatzes
NAME	text	Bezeichnung des Jobs
BESCHREIBUNG	text	nähere Beschreibung des Jobs
JOBNUMMER	integer	interne Jobnummer
BEGINN	timestamp	Beginn des Jobs
ENDE	timestamp	Ende des Jobs
FID_PROJEKTE	integer	Fremd-ID des Projektes aus Tabelle „projekte“
EINTRAGSDATUM	timestamp	Datum des Eintrages des Datensatzes in die Datenbank
XSTATUS	text	Status des Datensatzes (fertig, unfertig)

Abbildung 4.1.6: Datenbanktabelle „jobs“

Einzelnen Jobs werden, wie erwähnt, noch einmal in einzelne Tätigkeiten aufgeteilt, um eine feinere Strukturierung der zu erledigenden Arbeiten eines Projektes zu erhalten. Abbildung 4.1.7 stellt die Datenbanktabelle „taetigkeiten“ dar.
Wie auch in der Entität „jobs“ wird auch hier der „XSTATUS“ als Bearbeitungsstand der jeweiligen Tätigkeit benutzt.
Ein Beginn und eine Ende speichern die entsprechende zeitliche Etappe der Ausführung einer Tätigkeit.

Tabelle „taetigkeiten“		
Name	Typ	Beschreibung
ID	integer	eindeutige Identifikationsnummer des Datensatzes
NAME	text	Bezeichnung der Tätigkeit
BESCHREIBUNG	text	nähere Beschreibung der Tätigkeit
BEGINN	timestamp	Beginn der Tätigkeit
ENDE	timestamp	Ende der Tätigkeit
EINTRAGSDATUM	timestamp	Datum des Eintrages des Datensatzes in die Datenbank
XSTATUS	text	Status des Datensatzes (fertig, unfertig)

Abbildung 4.1.7: Datenbanktabelle „taetigkeiten“

Die Abbildung 4.1.8 zeigt die Datenbanktabelle „jobs_taetigkeiten“. In dieser so genannten Kreuztabelle werden die Tätigkeiten, welche zu einem bestimmten Job gehören, abgespeichert. Des Weiteren werden hier die Mitarbeiter, die eine bestimmte Tätigkeit ausführen sollen, geführt. Falls Tätigkeiten an externe Partner oder Lieferanten abgegeben werden sollen, wird das auch in dieser Tabelle vermerkt.
Alle diese Daten referenzieren über die jeweiligen Fremd-ID's ihre eigenen Datenbanktabellen.
Weiterhin wird ein Beginn und ein Ende sowie eine veranschlagte Zeit und ein veranschlagter Preis der auszuführenden Tätigkeit angegeben.

Tabelle „jobs_taetigkeiten“		
Name	Typ	Beschreibung
ID	integer	eindeutige Identifikationsnummer des Datensatzes
JOBNUMMER	text	eindeutige interne Jobnummer
FID_JOBS	integer	Fremd-ID des Jobs aus Tabelle „jobs“
FID_MITARBEITER	integer	Fremd-ID des Mitarbeiters aus Tabelle „mitarbeiter“
FID_TAETIGKEITEN	integer	Fremd-ID der Tätigkeit aus Tabelle „taetigkeiten“
FID_PARTNER	integer	Fremd-ID des Partners aus Tabelle „partner“
BEGINN	timestamp	Beginn der Tätigkeit
ENDE	timestamp	Datum des Endes der Tätigkeit
MINUTEN	integer	Anzahl der Minuten zur Ausführung der Tätigkeit
PREIS	float	veranschlagter Preis für Ausführung der Tätigkeit
EINTRAGSDATUM	timestamp	Datum des Eintrages des Datensatzes in die Datenbank
XSTATUS	text	Status des Datensatzes (fertig, nicht fertig)

Abbildung 4.1.8: Datenbanktabelle „jobs_taetigkeiten“

Die Datenbanktabelle „zeiterfassung“ (Abbildung 4.1.9) dient zur Speicherung der Arbeitszeiten eines Mitarbeiter, die er zur Ausführung einer einzelnen Tätigkeit benötigt hat. Der Mitarbeiter ist dabei über eine Fremd-ID aus der Tabelle „mitarbeiter“ und die Tätigkeit ist über eine Fremd-ID aus der Tabelle „jobs_taetigkeiten“ mit der Zeiterfassungstabelle verbunden.
Verbrauchte Ressourcen, die Anzahl Minuten und Bemerkungen geben einen genaueren Überblick über die vom Mitarbeiter ausgeführte Tätigkeit.

Tabelle „zeiterfassung“		
Name	Typ	Beschreibung
ID	integer	eindeutige Identifikationsnummer des Datensatzes
FID_MITARBEITER	integer	Fremd-ID des Mitarbeiters aus Tabelle „mitarbeiter“
FID_JOBSTAETIGKEITEN	integer	Fremd-ID der Tätigkeit aus Tabelle „jobs_taetigkeiten“
DATUM	timestamp	Datum der Ausführung der Tätigkeit
MINUTEN	integer	Bearbeitungsdauer der Tätigkeit
RESSOURCEN	text	dabei verbrauchte Ressourcen
BEMERKUNGN	text	Bemerkungen zur Ausführung der Tätigkeit
EINTRAGSDATUM	timestamp	Datum des Eintrages des Datensatzes in die Datenbank
XSTATUS	text	Status des Datensatzes (aktiv, gelöscht)

Abbildung 4.1.9: Datenbanktabelle „zeiterfassung“

4.2 Funktionen des Projektmanagement-Systems

4.2.1 Allgemeines Funktionsprinzip

Um das Projektmanagement-System überhaupt nutzen zu dürfen, muss sich ein Mitarbeiter erst am System anmelden.
Abbildung 4.2.1.1 zeigt einen solchen Anmeldebildschirm.

Abbildung 4.2.1.1: Anmeldung am Projektmanagement-System

Nach Eingabe der Zugangsdaten (Benutzer, Passwort) wird dieses zum Server übertragen. Dort wird zuerst der Benutzername mit den im Datenbanksystem befindlichen Benutzernamen (USER) aus der Tabelle „mitarbeiter“ verglichen. Ist keine Übereinstimmung vorhanden, erfolgt eine Fehlermeldung. Bei einer Übereinstimmung des eingegebenen Benutzernamens mit dem aus der Datenbank wird die komplette Datenbank-Zeile gelesen.

Aus dem im Anmeldebildschirm eingegebenen Passwort wird der SHA1-Hash gebildet und mit dem Passwort aus der Datenbank, welches auch als SHA1 abgelegt wurde, verglichen. Stimmen beide Hashes überein, startet der Server eine Session, in der die ID des Mitarbeiter-Datensatzes und die Rechte des Mitarbeiters gespeichert werden. Dies hat den Vorteil, dass keine Datenbankoperation mehr nötig ist, wenn diese Daten gebraucht werden. Zum Beispiel ist im Bereich der Zeiterfassung die ID des Mitarbeiters in die Datenbanktabelle „zeiterfassung“ einzutragen. Durch die in der Session befindliche Mitarbeiter-ID wird somit eine Datenbank-Operation eingespart.

Sicherheitstechnisch sind Sessions als sicher anzusehen, da nur eine Kommunikation mit einem vom Session-Management des HTTP-Servers erlaubten Client erlaubt ist. Das Session-Prinzip funktioniert folgendermaßen: Der Server erzeugt eine 32-stellige alphanumerische Session-ID, speichert diese und sendet sie zum Client, die dort entweder als Cookie gespeichert oder bei jedem Serverzugriff mit übertragen wird. So ist gewährleistet, dass nur vom Server authentifizierte Clients Zugang zum session-geschützten Bereich des Servers haben.
Als einzigen sicherheitskritischen Vorgang kann die Übertragung des im Anmeldebildschirm eingegebenen Passwortes angesehen werden, da dieses unverschlüsselt zum HTTP-Server übertragen wird. Eine SSL-Verschlüsselung bietet aber die Möglichkeit, dieses Sicherheitsproblem zu lösen und eine verschlüsselte Kommunikation zwischen Server und Client zu gewährleisten.

4.2.2 Funktionen der Mitarbeiterverwaltung

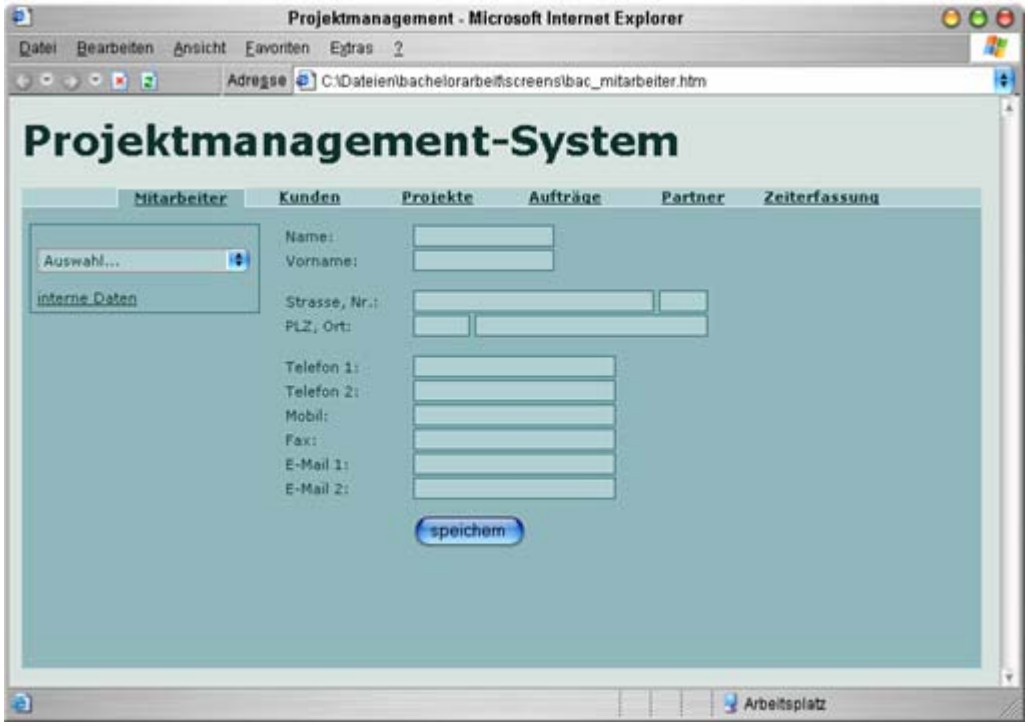


Abbildung 4.2.2.1: Programmoberfläche der Mitarbeiterverwaltung (persönliche Daten)

Abbildung 4.2.2.1 stellt eine Programmoberfläche, welche zur Verwaltung der Mitarbeiter Verwendung findet, dar. In der Darstellung ist dies der Karteireiter „Mitarbeiter“.

Im linken Drop-Down-Menü wird ein im System gespeicherter Mitarbeiter ausgewählt. Seine Daten werden in den vorhandenen Formularfeldern angezeigt, so dass dort eine Änderungsmöglichkeit besteht. Des Weiteren gewährleistet dieses Drop-Down-Feld, neue Mitarbeiter ins System aufzunehmen, wie auch Mitarbeiterdaten zu löschen. Das Löschen ist deshalb als Extra-Menüpunkt aufgenommen worden, um Fehlbedienungen des Systems so gering wie möglich zu halten.

Wurden alle Textfelder ausgefüllt, werden zunächst die Angaben auf Korrektheit überprüft, d. h. ob nicht zum Beispiel im Feld „Telefon 1“ ein Buchstabe enthalten ist. Falls Falscheintragungen vorhanden sind, weisen Warnhinweise auf diesen Umstand hin. Sind alle Felder korrekt ausgefüllt, nimmt das System Kontakt zum Datenbank-System auf und testet, ob schon identische Datensätze enthalten sind. Ist dies der Fall, wird der Benutzer gefragt, ob er den vorhandenen Datensatz überschreiben oder die eingegebenen Daten nochmals ändern möchte. Bei Nichtvorhandensein eines identischen Datensatzes wird eine neuer Satz ins Datenbank-System aufgenommen.
Die Kontaktaufnahme zum Datenbank-System erfolgt über die PHP-Erweiterung, die an eine bestimmte Schnittstelle (CGI) des HTTP-Servers gebunden ist. Dabei baut die PHP-Funktion „mysql_connect('host','user','pass')“ die eigentliche Datenbank-Verbindung auf. Nach erfolgreichem Login am Datenbank-System wird nun die Tabelle „mitarbeiter“ mittels der PHP-Funktion „mysql_select_db('mitarbeiter')“ selektiert. Der SQL-Query „select * from mitarbeiter where XSTATUS='aktiv'“ liest alle Datensätze aus der Datenbanktabelle „mitarbeiter“, deren XTATUS „aktiv“ ist. Die PHP-Funktion mysql_fetch_array() führt diese Abfrage aus. Die Datensätze befinden sich nun in einem so genannten assoziativen Array, d. h. über die Spaltennamen der Datenbanktabelle ('NAME', 'VORNAME', etc.) wird auf die Abfrage-Ergebnisse zugegriffen.
Die Funktion „mysql_close()“ schließt die vorhandene Datenbank-Verbindung.

Weiterhin befindet sich unter dem Drop-Down-Menü aus Abbildung 4.2.2.1 ein Link „interne Daten“, der eine Verwaltung der internen und auch sensiblen Daten eines Mitarbeiters bietet.

Abbildung 4.2.2.2 zeigt nun diese Programmoberfläche, wie sie zur Verwaltung der internen Daten verwendet wird.

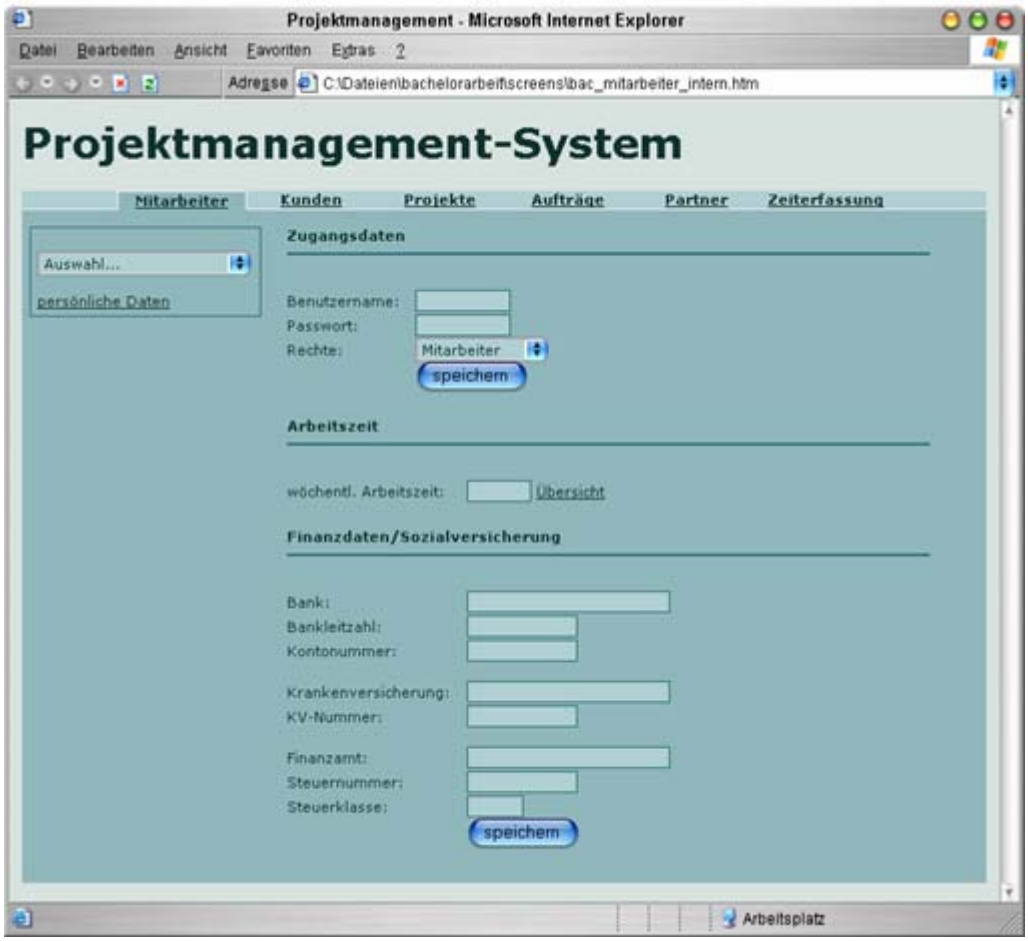


Abbildung 4.2.2.2: Programmoberfläche der Mitarbeiterverwaltung (interne Daten)

Im oberen Bereich „Zugangsdaten“ wird ein Management der Zugangsdaten eines Mitarbeiters gewährleistet. Auch werden dort seine Rechte, die er im Projektmanagement-System besitzt, definiert.

Nach Auswahl eines Mitarbeiters oder Übernahme des aktuellen Mitarbeiters von der vorherigen Bildschirmmaske „persönliche Daten“ werden die vorhandenen Textfelder mit den im Datenbank-System befindlichen Daten aus der Tabelle „mitarbeiter“ befüllt.

Die Konnektierung des Datenbank-Systems erfolgt mit Hilfe der PHP-Funktionen mysql_connect(), mysql_select_db(), mysql_fetch_array() und mysql_close().
Da das Passwort als SHA1-Hash gespeichert ist und somit eine Entschlüsselung des Passwortes unmöglich ist, werden hier nur 10 Sternchen, hinter denen eine fest vorgegebene Zeichenfolge steht, angezeigt.
Bei Änderung der Zugangsdaten oder Rechte erfolgt zunächst ein Vergleich, ob der Text im Passwort-Feld geändert wurde - eine vorherige Vergabe eines standardisierten Strings ist ja erfolgt, die einen Vergleich ermöglicht. Ist dies der Fall, so wird aus dem neuen Passwort der SHA1-Hash erzeugt und dieser als neues Passwort in die Datenbank eingetragen. Zur Generierung des SHA1-Hashes kommt die PHP-Funktion sha1() zum Einsatz. Falls keine Änderungen am Passwort erfolgt sind, beispielsweise bei Bearbeitung der Rechtezuteilung, wird das Eintragen des Passwortes in die Datenbank übergangen und nur die geänderten Felder eingetragen.
Das Absenden des Eingabeformulars geschieht über die so genannte POST-Methode, die im HTTP-Protokoll implementiert ist.
Da hinter jedem Textfeld des Eingabeformulars eine Variable steht, erfolgt der serverseitige Zugriff auf die Daten über diese Variablen. Nachdem die Variablenbelegungen auf Korrektheit und Plausibilität überprüft wurden, wird mit Hilfe der SQL-INSERT-Anweisung der Datensatz eingefügt, bzw. mit Hilfe der SQL-UPDATE-Anweisung der Datensatz verändert.

Im mittleren Teil „Arbeitszeit“ wird seine wöchentliche Arbeitszeit im System hinterlegt. Der Platzhalter „Übersicht“ zeigt in einem neuen Fenster alle bisher erbrachten Arbeitsstunden. Diese werden aus der Datenbanktabelle „zeiterfassung“ ermittelt. Dort sind die ID des Mitarbeiters, die ID aus der Tabelle „jobs_taetigkeiten“ als „FID_JOBSTAETIGKEITEN“ und die Zeit, die zur Ausführung des Jobs/der Tätigkeit benötigt wurden, hinterlegt, wodurch sich eine solche Übersicht erzeugen lässt.

Im Abschnitt „Finanzdaten/Sozialversicherung“ werden alle finanz- und sozialversicherungsrelevanten Daten eines Mitarbeiters verwaltet. Dazu erfolgt auch hier wieder eine Kommunikation mit der Datenbanktabelle „mitarbeiter“.

Warnungen beim Speichern geben Hinweise auf nicht ausgefüllte bzw. fehlerhaft ausgefüllte Felder.

4.2.3 Funktionen der Kundenverwaltung

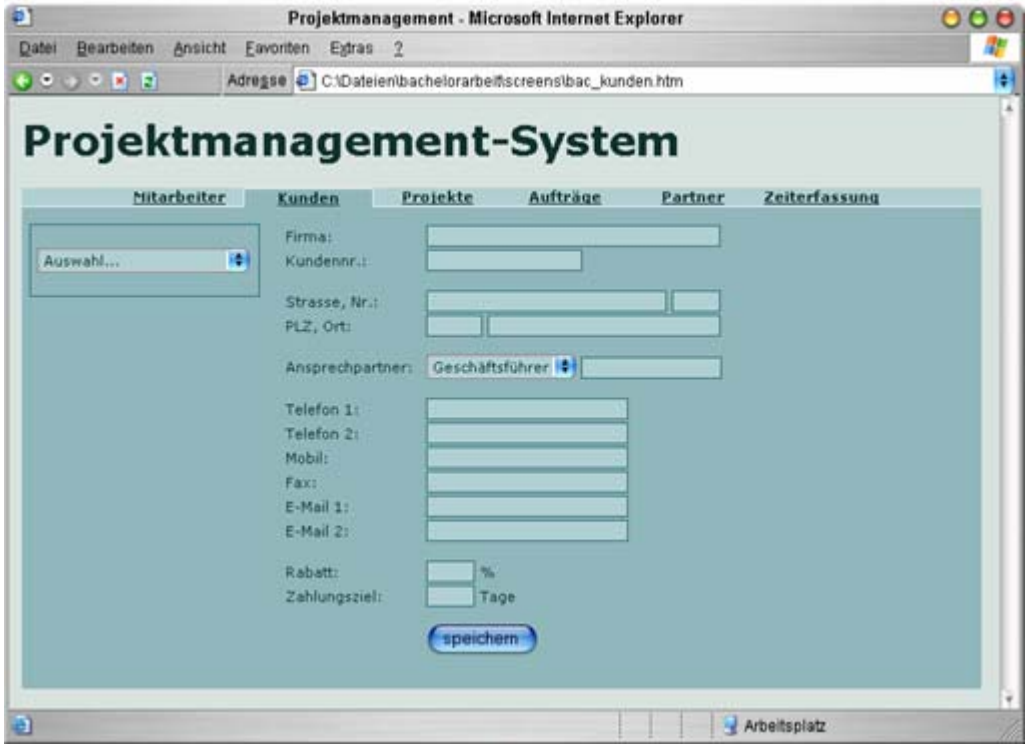


Abbildung 4.2.3.1: Programmoberfläche der Kundenverwaltung

Abbildung 4.2.3.1 zeigt die Programmoberfläche, die zur Verwaltung der Kundendaten Verwendung findet. In der Darstellung ist dies der Karteireiter „Kunden“.

Das Drop-Down-Feld im linken oberen Bereich dient zur Kundenauswahl. Nach Verbindungsaufnahme mit der Datenbank, mittel der PHP-Funktionen mysql_connect() und mysql_select_db(), werden aus der Datenbanktabelle „kunden“ diejenigen Datensätze ausgelesen, deren XSTATUS auf „aktiv“ steht, so dass alle aktiven Kunden in diesem Drop-Down-Feld erscheinen, welche dort zur Auswahl stehen. Der SQL-Query „select * from kunden where XSTATUS=’aktiv’“, welcher mit Hilfe der PHP-Funktion mysql_fetch_array() ausgeführt wird, realisiert dies.

Nach Selektion eines Kunden erfolgt ein Übertragen der im Drop-Down-Feld hinterlegten ID des ausgewählten Kunden zum Server. Dort wird, nach vorheriger Konnektierung des Datenbank-Systems, der Datensatz des mit der ID spezifizierten Kunden ausgelesen. Dies realisiert der SQL-Query „select * from kunden where XSTATUS='aktiv' & ID='$id_kunden'“, wobei '$id_kunden' die vom Client übertragene Kunden-ID ist. Danach werden die so aus der Datenbank erhaltenen Daten durch die vorhandenen Texteingabefelder angezeigt, wo eine Bearbeitung möglich ist. Des Weiteren gewährleistet das Drop-Down-Feld, neue Kundendaten anzulegen wie auch vorhandene Kundendaten zu löschen.

Zu den Kundendaten gehören alle Geschäftsverbindungsdaten, wie Firma, Adresse, Telefonnummern, eine Faxnummer sowie E-Mail-Adressen.

Eine eindeutige Kundennummer spezifiziert einen Kunden.

Um eine persönliche Verbindung zum Kunden herzustellen, wird zudem ein Ansprechpartner und seine Position im Unternehmen vermerkt.

Weiterhin werden ein mit dem Kundenunternehmen vereinbarter Rabatt, sowie ein vereinbartes Zahlungsziel, innerhalb dessen offene Rechnungen beglichen werden müssen, gespeichert. Ist kein Rabatt oder ein Zahlungsziel in eine Rechnung aufzunehmen, wird das entsprechende Texteingabefeld leer gelassen.

Sollen Kundendaten gelöscht werden, so ist zunächst dieser Kunde im Drop-Down-Feld auszuwählen. Seine Daten werden, wie erwähnt, in den vorhandenen Textfeldern angezeigt. Ist dies erfolgt, wird zum Löschen „Kunde löschen...“ im Drop-Down-Feld ausgewählt, wo nach einer weiteren Bestätigung (zur Sicherheit) die Daten eines Kunden gelöscht sind. Im Speziellen ist der Kunde nur deaktiviert worden, indem das Datensatz-Feld „XSTATUS“ auf „inaktiv“ gesetzt wurde. Die Kundendaten selbst sind im System noch vorhanden. Das Deaktivieren erfolgt mit Hilfe des SQL-Querys „update kunden set XSTATUS=’inaktiv’ where ID=’$id_kunden’“, der, nach Verbindungsaufnahme zum Datenbank-System, mittels der PHP-Funktion mysql_fetch_array() zur Datenbank gesendet wird.

Dieses Vorgehen kommt nicht nur hier in der Kundenverwaltung so zum Einsatz, sondern funktioniert ebenfalls im gesamten Projektmanagement-System. Eine Ausnahme dabei bildet die Auftragsverwaltung, da dort der XSTATUS den Bearbeitungsstand einzelner Jobs oder Tätigkeiten definiert.

4.2.4 Funktionen der Projektverwaltung

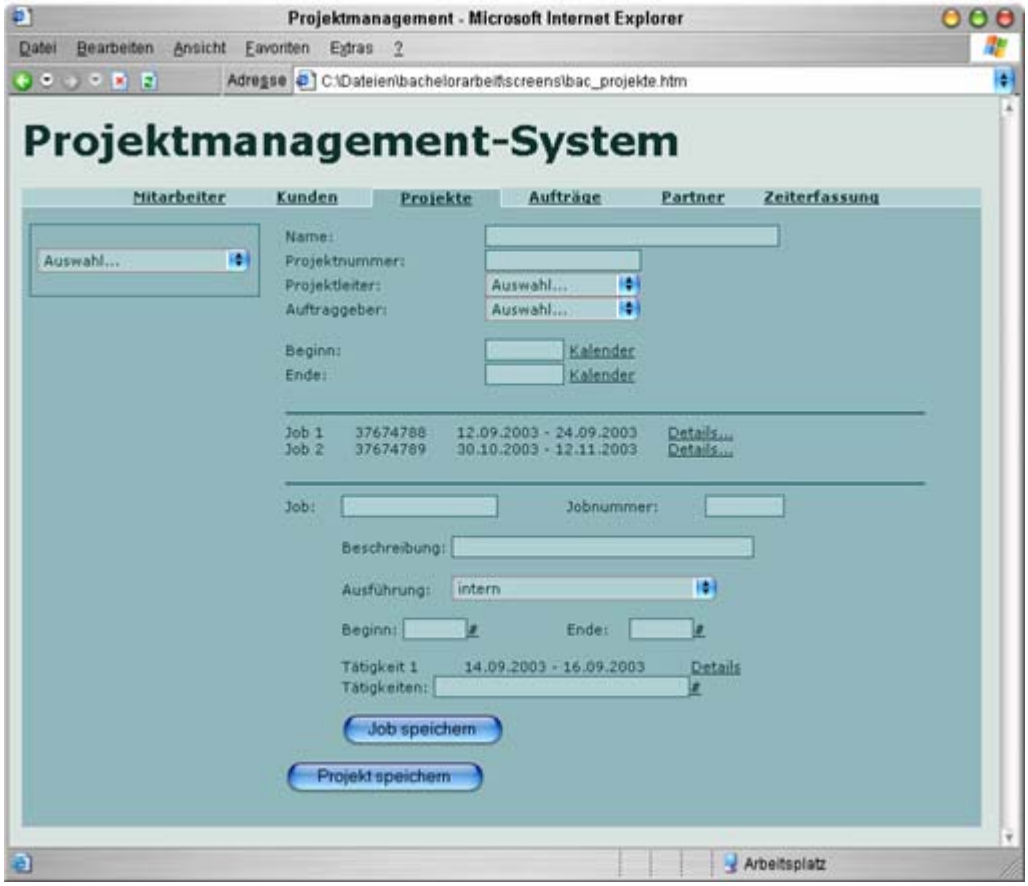


Abbildung 4.2.4.1: Programmoberfläche der Projektverwaltung

Die Verwaltung der Projektdaten (Karteireiter „Projekte“) erfolgt über eine Programmoberfläche, wie sie in Abbildung 4.2.4.1 dargestellt ist.

Die Selektion eines Projektes erfolgt durch ein Drop-Down-Feld, wo nach Verbindungsaufnahme mit der Datenbank die aktiven Projekte aus der Datenbanktabelle „projekte“ ausgelesen werden. Der dies realisierende SQL-Query heißt hier „select * from projekte where XSTATUS='aktiv'“.

Danach werden die Textfelder mit den im System gespeicherten Daten befüllt, indem wiederum Kontakt mit der Datenbank aufgenommen wird. Ein „select * from projekte where XSTATUS='aktiv' & ID='$id_projekte'“, welches mit der PHP-Funktion mysql_fetch_array() ausgeführt wird, selektiert den im Drop-Down-Feld spezifizierten Projekt-Datensatz aus der Datenbank. Auf die einzelnen Datensatzelemente kann über ein assoziatives Array zugegriffen werden, das ja mit der PHP-Funktion mysql_fetch_array() erzeugt wurde.
Das genaue Vorgehen ist dies, dass zuerst der SQL-Query in eine Variable gespeichert wird:

```
$query_projekte=mysql_query(„select * from projekte where XSTATUS='aktiv' & ID='$id_projekte'“);
```

Danach erfolgt das Ausführen der SQL-Anfrage und Speichern des erhaltenen Datensatzes in das assoziative Feld:

```
$arr_projekte=mysql_fetch_array($query_projekte);
```

Der Zugriff auf die einzelnen Elemente des Datensatzes erfolgt über den Feldnamen, als Argument wird der Spaltennamen des Datensatzelementes angegeben. So gibt beispielsweise ein $arr_projekte[„PROJEKTLEITER“] den Namen eines in der Datenbank befindlichen Namen des Projektleiters eines Projektes zurück.

Beim Neuanlegen eines Projektes wird dem Projekt ein Name gegeben. Eine eindeutige Projektnummer spezifiziert das Projekt. Nach dem Festlegen eines Projektleiters, der zu dem Mitarbeiterkreis gehören muss, wird der Auftraggeber, für den das Projekt ausgeführt wird, selektiert. Im Feld „Projektleiter“ werden alle aktiven Mitarbeiter, die zudem noch die Rechte eines Projektleiters besitzen, aus der Datenbanktabelle „mitarbeiter“ ausgelesen. Des Weiteren ist in der Datenbanktabelle „projekte“ eine Spalte „FID_KUNDEN“ enthalten, die als Fremd-Schlüssel denjenigen Datensatz referenziert, dessen ID mit der genannten FID_KUNDEN übereinstimmt. Da in der Datenbanktabelle „kunden“ die potentiellen Auftraggeber gespeichert sind, ist somit sichergestellt, dass ein Projekt zu einem bestimmten Auftraggeber gehört.

Sollte der Kunde/Auftraggeber noch nicht im System enthalten sein, wird über das Auswahl-Menü zur Kundenverwaltung verwiesen, wo ein neuer Auftraggeber ins System aufgenommen wird.

Ein Beginn und ein Ende grenzen die zeitliche Realisierung ein.

Projekte werden zunächst in einzelne Jobs aufgeteilt – Daten, wie Beginn, Ende, eine nähere Beschreibung, konkretisieren einen Job. Diese Job-Daten werden in eine separate Datenbanktabelle „jobs“ aufgenommen. Damit ein einzelner Job zu einem Projekt zugeordnet werden kann, wird die Projekt-ID als Fremdschlüssel „FID_PROJEKTE“ in die Tabelle „jobs“ mit aufgenommen. Danach ist ein einzelner Job wiederum in Tätigkeiten zu separieren. Die Eingabe der eine Tätigkeit näher spezifizierenden Daten erfolgt hier über einen Platzhalter „#“. Dabei wird ein neues Fenster geöffnet, in dem dann die restlichen Textfelder, wie Beginn, Ende, Beschreibung, auszufüllen sind. Nach dem Abspeichern der neuen Tätigkeit in diesem Edit-Fenster wird die hinzugefügte Tätigkeit im Hauptfenster aufgelistet. Für das Speichern der Daten von Tätigkeiten wird wiederum die Datenbank konnektiert und diese Daten in der Tabelle „taetigkeiten“ abgelegt. Der SQL-Query, der das Abspeichern der Tätigkeit realisiert, lautet „insert into taetigkeiten(NAME, BESCHREIBUNG, BEGINN, ENDE, EINTRAGSDATUM, XSTATUS) values (’$name’, ’$beschreibung’, ’$beginn’, ’$ende’, ’$eintragsdatum’, ’unfertig’)“. Dabei referenzieren die Variablen $name, $beschreibung, etc. die ausgefüllten Textfelder der grafischen Oberfläche.

Soll ein Job von einem externen Unternehmen, welches dann als Auftragnehmer auftritt, ausgeführt werden, so wird dies im Datenfeld „Ausführung“ vermerkt. Dieses beinhaltet alle im System befindlichen Datensätze aus der Partner-/Lieferantenverwaltung. Bei Nichtvorhandensein eines externen Partners wird hier zur Partner-/Lieferantenverwaltung verwiesen. In der Datenbanktabelle „jobs_taetigkeiten“ sind alle zur Projektverwaltung gehörenden Datenbanktabellen über ihre jeweiligen ID's als Fremdschlüssel verbunden. So ist zum Beispiel dort vermerkt, welche Tätigkeiten zu welchen Jobs gehören und von welchem Lieferanten dieser Job

evtl. ausgeführt wird. Ist die Ausführung nicht an einen externen Partner gebunden, so wird im Datensatz-Feld „FID_PARTNER“ nicht die ID des Partners gespeichert, sondern es ist eine Marke, beispielsweise „intern“, einzutragen.

Da diese Einzel-Jobs bzw. Einzel-Tätigkeiten in der Angebotserstellung bzw. Rechnungslegung erscheinen, muss hier schon bei der Namensgebung eine klare und aussagekräftige Ausdrucksweise gegeben werden.

Nach dem Abspeichern des gesamten Jobs (Senden zum Server, Speichern in Datenbank) erscheint dieser in der Jobliste.

Ein Klicken der Schaltfläche „Projekt speichern“ finalisiert das Anlegen eines Projektes, indem alle in den Textfeldern befindlichen Daten zum Server gesendet werden und dort in die Datenbanktabelle „projekte“ gespeichert werden.

Die Schaltfläche „Details“ bietet die Möglichkeit, vorhandene Job- bzw. Tätigkeitsdaten zu bearbeiten.

Die Platzhalter „#“ finden im Bereich der Projektverwaltung mehrfach Anwendung, die aber immer unterschiedliche Funktionalitäten realisieren. Diese erschließen sich aber im logischen Zusammenhang mit den verschiedenen Eingabefeldern. So dient beispielsweise der Platzhalter „#“ hinter den Feldern „Beginn“ und „Ende“ der einfacheren Möglichkeit der Eingabe eines Datums, da sich beim Anklicken ein Fenster öffnet, in dem ein Kalender dargestellt ist, so dass dort das genaue Datum ausgewählt wird.

4.2.5 Funktionen der Auftragsverwaltung

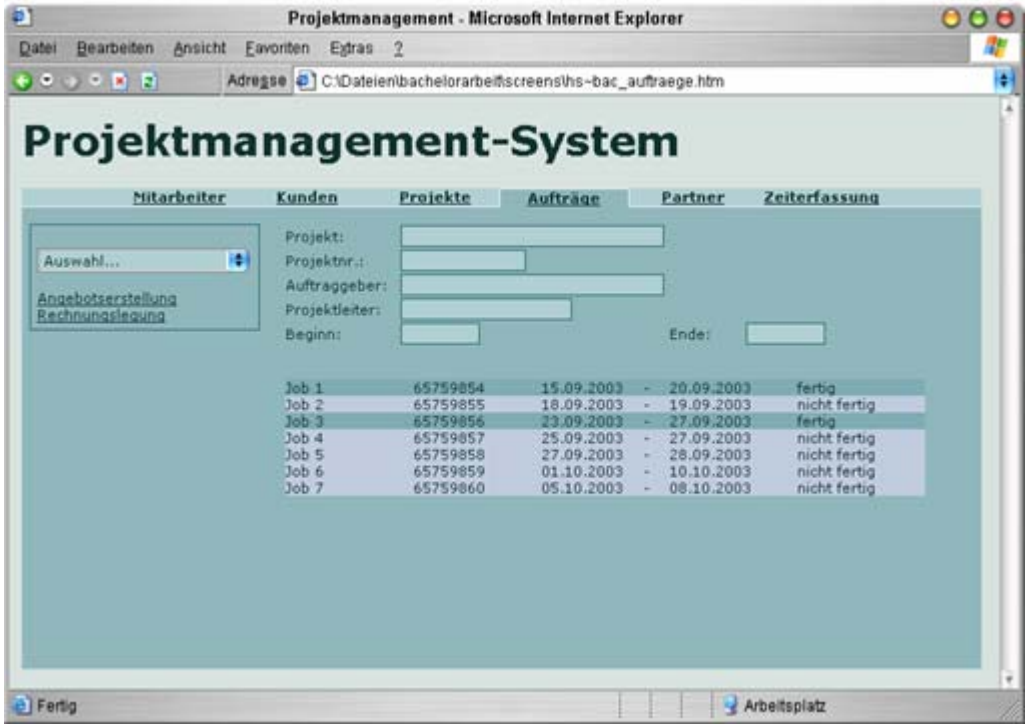


Abbildung 4.2.5.1: Programmoberfläche der Auftragsverwaltung (Übersicht)

Abbildung 4.2.5.1 stellt die Übersichtsseite der Auftragsverwaltung dar. In der grafischen Bedienoberfläche ist dies durch den Karteireiter „Aufträge“ dargestellt.

Im linken oberen Drop-Down-Menü wird ein vorhandenes Projekt ausgewählt, danach werden die dazugehörigen Textfelder mit den im Projektmanagement-System befindlichen Datensätzen befüllt, und es erfolgt eine Auflistung aller zum Projekt gehörenden Einzel-Jobs. Dazu wird die Datenbank konnektiert (mysql_connect(), mysql_select_db()) und die jeweiligen Daten aus der Datenbanktabelle „projekte“ ausgelesen (select * from projekte where XSTATUS=’aktiv’ & ID=’$id_projekte’).

Das Extrahieren der Jobliste erfolgt über die Entität „jobs“. Da in dieser Tabelle die Spalte „FID_PROJEKTE“ ein zugehöriges Projekt referenziert, ist das Erfassen der zu einem Projekt gehörenden Einzel-Jobs möglich. Genau geschieht dies mit Hilfe des SQL-Querys „select * from jobs where FID_PROJEKTE='$id_projekte'“, wobei $id_projekte die ID des im Drop-Down-Feld ausgewählten Projektes ist.

Zu dieser Job-Liste gehören die Bezeichnung des Jobs, eine Jobnummer, der zeitliche Beginn und das Ende des Jobs sowie der aktuelle Status (fertig, nicht fertig) der Ausführung des jeweiligen Einzel-Jobs. Da in der Datenbank sämtliche zeitliche Daten, wie Beginn und Ende, als Unix-Timestamp[2] gespeichert sind, muss aus diesem ein für den Benutzer verständliches Datenformat erzeugt werden. Dafür stehen 2 Strategien zur Verfügung: entweder das Datenbank-System wandelt das Datenformat direkt mit Hilfe eines SQL-Query's (select DAYOFMONTH(BEGINN) as tag, MONTH(BEGINN) as monat, YEAR(BEGINN) as jahr from jobs where FID_PROJEKTE='$id_projekte') um, oder der Timestamp wird als solcher ausgelesen, und die Skriptsprache, welche als Server-Erweiterung eingebunden ist, formatiert das Datum. Diese Formatierung realisiert die PHP-Funktion date(), im Speziellen:

```
$datum=date(„d“,$arr_jobs[BEGINN]).“.“;
$datum.=date(„m“,$arr_jobs[BEGINN]).“.“;
$datum.=date(„y“,$arr_jobs[BEGINN]);
```

Die Argumente „d“, „m“, „y“ geben jeweils den Tag, Monat und das Jahr aus dem im Feld „$arr_jobs[]“ befindlichen Timestamp zurück.

[2] Anzahl Sekunden seit 1.1.1970, 00:00:00 Uhr, GMT

Da es beispielsweise bei verschiedenen PHP-Versionen zu Wandlungsfehlern kam, ist eine direkte Formatierung per SQL-Query zu empfehlen.

Ein Klick in eine Zeile der Job-Liste bringt in einem neuen Fenster genauere Daten der Einzelposten hervor (s. Abb. 4.2.5.2).

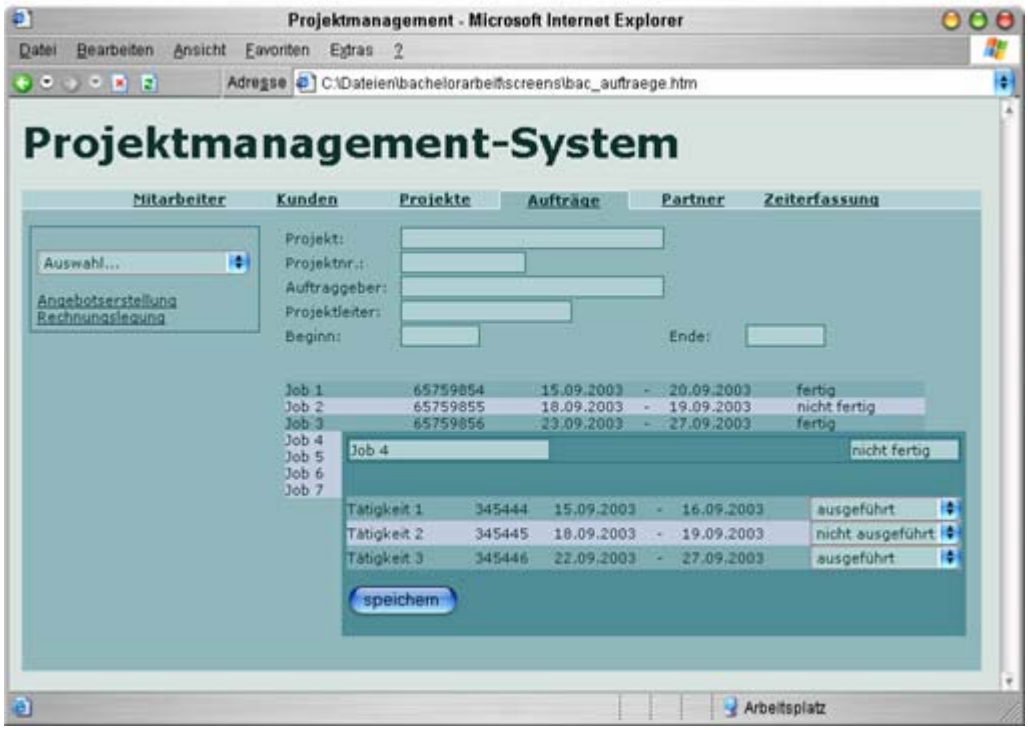


Abbildung 4.2.5.2: Programmoberfläche der Auftragsverwaltung (Detailansicht)

Wie in Abbildung 4.2.5.2 dargestellt, kann im neu geöffneten Fenster der Ausführungs-Status aller zu einem Einzel-Job gehörenden Tätigkeiten geändert werden. Das Extrahieren der zu einem Job gehörenden Tätigkeiten erfolgt durch Auslesen der Datenbanktabellen „taetigkeiten“ und „jobs_taetigkeiten“.

Zuerst, da die ID eines Jobs vorhanden ist, werden alle Datensätze aus der Entität „jobs_taetigkeiten“ extrahiert, deren FID_JOBS der schon bekannten ID des Jobs entspricht. Dies realisiert der SQL-Query: „select * from jobs_taetigkeiten where FID_JOBS='$id_jobs'“. Die Ergebnisdaten sind mit Hilfe der PHP-Funktion „mysql_fetch_array()“ in ein assoziative Feld abgelegt worden.
In den erhaltenen Datensätzen sind die Tätigkeiten durch ihre Fremdschlüssel „FID_TAETIGKEITEN“ vertreten, so dass nach einem weiteren Auslesen aus der Entität „taetigkeiten“ und einem Vergleich der ID der Tätigkeit mit der „FID_TAETIGKEITEN“ der Tabelle „jobs_taetigkeiten“ die Daten derjenigen Tätigkeiten ausgelesen werden, die zu einem bestimmten Job gehören. Der SQL-Query dazu lautet: „select * from taetigkeiten where ID='$arr_jobs[“FID_TAETIGKEITEN“]'“. Dabei ist „$arr_jobs[]“ das assoziative Array aus der vorherigen SQL-Abfrage, die die einzelnen Fremdschlüssel-ID's aus der Datenbanktabelle „jobs_taetigkeiten“ extrahiert.

Der Bearbeitungsstand einer Tätigkeit („ausgeführt“, „nicht ausgeführt“) wird in der Datenbank durch die Spalte „XSTATUS“ definiert. Nach Speicherung der Bearbeitungsstände werden diese in der Datenbank aktualisiert. Sind alle Tätigkeiten als „ausgeführt“ markiert, so gilt der übergeordnete Job auch als ausgeführt, und in der Datenbanktabelle „jobs“ muss aus diesem Grund der „XSTATUS“ auf „fertig“ gesetzt werden. Der SQL-Befehl „update jobs set XSTATUS='fertig' where ID='$id_jobs'“ setzt in dem Datensatz, dessen ID der aktuellen Job-ID ($id_jobs) entspricht, den „XSTATUS“ auf „fertig“.

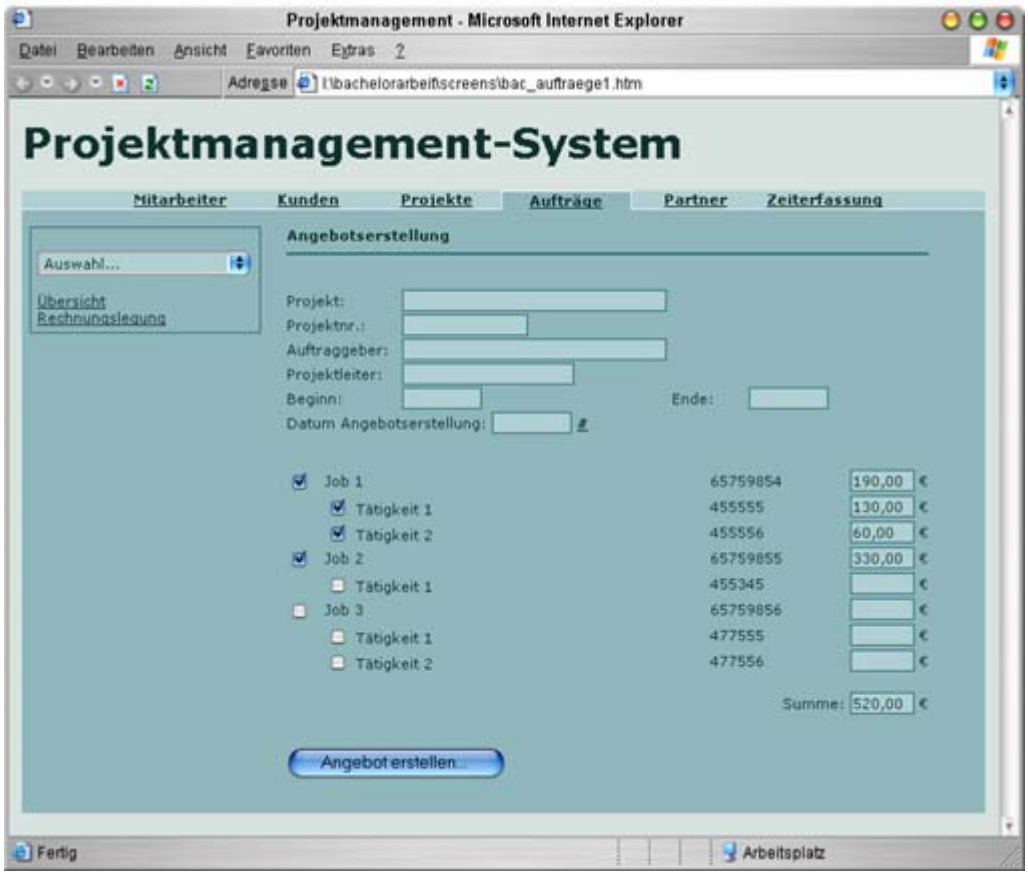


Abbildung 4.2.5.3: Programmoberfläche der Auftragsverwaltung (Angebotserstellung)

Abbildung 4.2.5.3 zeigt die Programmoberfläche der Angebotserstellung des Themenbereiches „Auftragsverwaltung“ (Karteireiter Aufträge). Damit ist eine automatisierte Generierung eines Angebotes für ein bestimmtes Projekt möglich. Im oberen Teil werden die allgemeinen Daten des Projektes dargestellt, indem die Datenbanktabelle „projekte“ ausgelesen wird. Im Weiteren erfolgt eine Auflistung aller zugehörigen Jobs inkl. aller Einzel-Tätigkeiten, die aus den Datenbanktabellen „jobs“ und „taetigkeiten“ stammen.

Das programmtechnische Vorgehen hierbei ist das gleiche, wie es im Text zur vorherigen Abbildung 4.2.5.2 beschrieben ist.

Über Fremdschlüssel in der Datenbanktabelle „jobs_taetigkeiten“ sind die jeweiligen Datensätze untereinander verbunden, so dass eine Zugehörigkeit einer Tätigkeit zu einem bestimmten Job definiert ist.

Die Programmoberfläche bietet nun ein Eintragen eines Preises für die Ausführung einer Tätigkeit bzw. eines Jobs an, welcher auf einem Angebot erscheint. Dabei werden nur die Posten im Angebot zur Anzeige gebracht, denen ein Preis zugeordnet wurde.

Nach einem Klick auf die Schaltfläche „Angebot erstellen...“ erfolgt eine Aktualisierung der Job- bzw. Tätigkeits-Datensätze, für die ein Preis festgelegt wurde. Der SQL-Befehl für das Eintragen des Preises eines Jobs dazu lautet: „update jobs set PREIS=’$preis’ where ID=’$id_jobs’“ Dabei ist „$id_jobs“ die aktuelle ID des zu bearbeitenden Datensatzes. Alle nicht markierten Posten werden in der Datenbank nicht geändert und erscheinen auch nicht auf einer späteren Rechnung. Nach diesem Vorgang öffnet sich ein neues Fenster (s. Abb. 4.2.5.4), in dem die aktivierten und mit einem kalkulierten Preis versehenen Einzel-Jobs und Einzel-Tätigkeiten dargestellt sind. Ein „select * from jobs where FID_PROJEKTE=’$id_projekt’ & PREIS<>0“ realisiert dies, indem nur diejenigen Datensätze ausgegeben werden, die zum aktuellen Projekt gehören (FID_PROJEKTE=’$id_projekt’) und deren Preis ungleich null ist (PREIS<>0).

Außerdem bietet ein mehrzeiliges Texteingabefeld die Möglichkeit, allgemeinen Angebotstext zu erstellen. Abschließend erfolgt ein Ausdruck, Adressdaten des Kunden werden automatisch in das Angebot eingefügt, indem der entsprechende Datensatz aus der Datenbanktabelle „kunden“ extrahiert wird, der wiederum über einen Fremdschlüssel „FID_KUNDEN“ mit der Entität „projekte“ verbunden ist.

Eine Bearbeitung dieser Daten ist möglich.

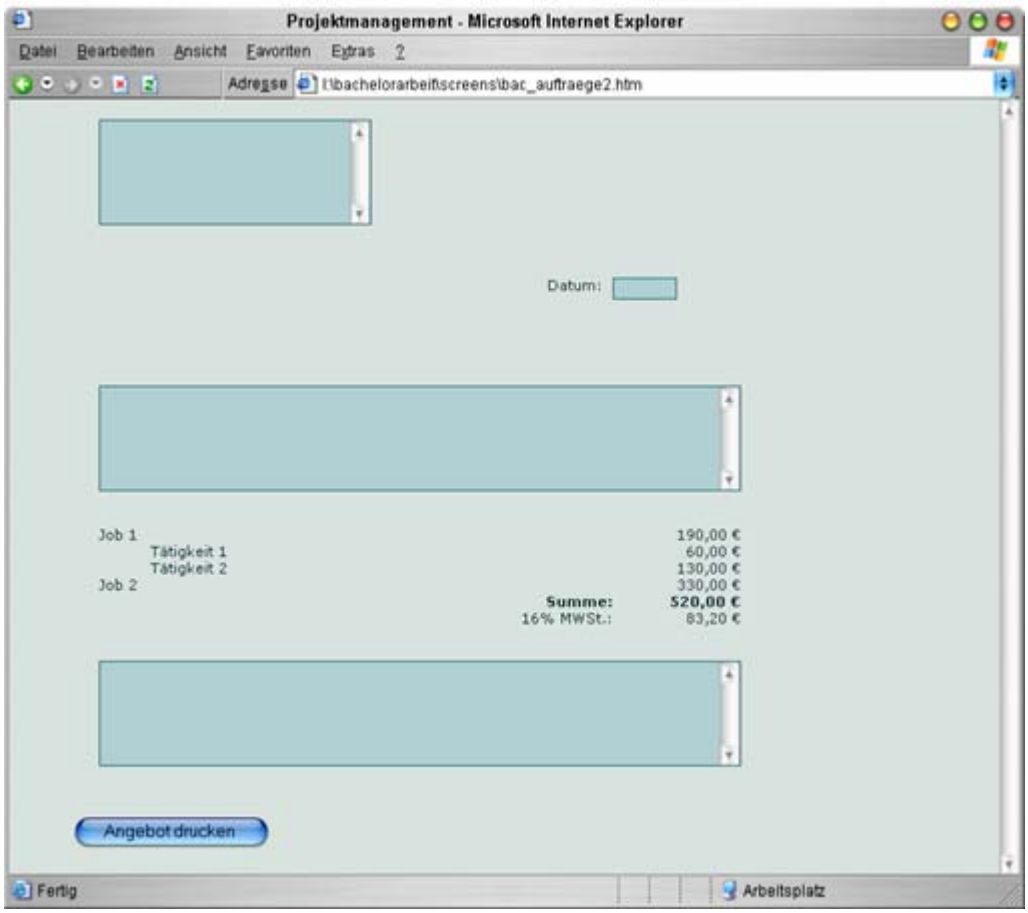


Abbildung 4.2.5.4: Programmoberfläche der Auftragsverwaltung (Angebotserstellung-Druck)

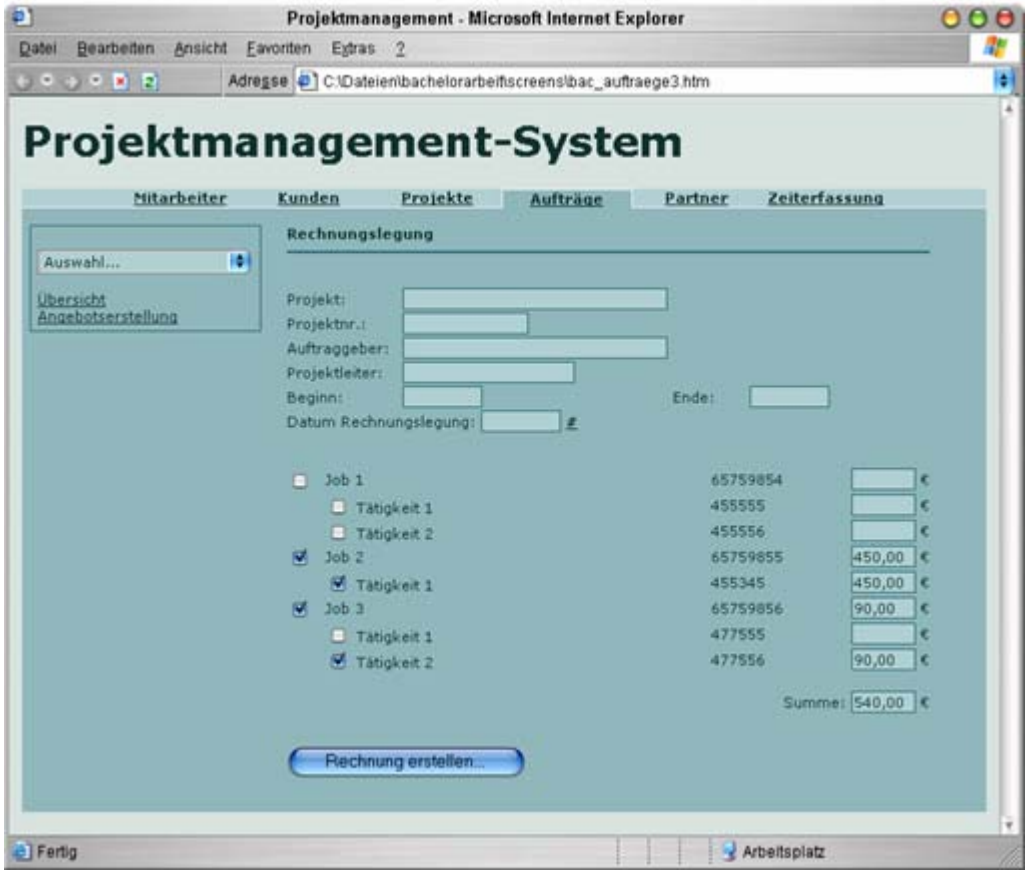


Abbildung 4.2.5.5: Programmoberfläche der Auftragsverwaltung (Rechnungslegung)

In der „Rechnungslegung“ des Themenbereiches „Auftragsverwaltung“ erfolgt eine automatische Generierung einer Projekt-Rechnung mit den Daten eines zu Grunde liegenden Angebotes, wie in Abbildung 4.2.5.5 ersichtlich ist.

Da bei der Angebotserstellung die Preise für die Einzel-Jobs und Einzel-Tätigkeiten ins System aufgenommen wurden, werden diese nun hier noch einmal aufgelistet, und es besteht die Möglichkeit, diese nachträglich zu ändern.

Im Speziellen werden hier nur diejenigen Datensätze aus den Datenbanktabellen „jobs“ und „taetigkeiten“ ausgelesen, die im Feld „PREIS“ einen entsprechenden Wert enthalten. Der SQL-Query, der eine solche Auflistung der Jobs ermöglicht, lautet: „select * from jobs where FID_PROJEKTE='$id_projekt' & PREIS<>0“, wobei „$id_projekt“ die ID des aktuellen Projektes ist.
Posten ohne einen Preis wurden in einer Angebotserstellung nicht berücksichtigt und fließen daher auch nicht in eine Rechnung mit ein.
Für Jobs oder Tätigkeiten, die in einem Angebot kalkuliert worden sind, aber auf einer Rechnung nicht erscheinen sollen, besteht nun hier noch die Möglichkeit, diese durch Entfernen des Markierungshäkchens (s. Abb. 4.2.5.5) von der Anzeige auf einer Rechnung auszuschließen.
Nach einem Klick auf die Schaltfläche „Rechnung erstellen...“ öffnet sich - ähnlich der Angebotserstellung - ein Fenster, in dem die Möglichkeit besteht, nach dem Erstellen eines allgemeinen Rechnungstextes, diese Rechnung auszudrucken.
Adressdaten werden automatisch in die Rechnung eingefügt, indem die Datenbank konnektiert wird und aus der Tabelle „kunden“ der entsprechende Aufraggeber ausgelesen wird, dessen ID mit der „FID_KUNDEN“ aus der Tabelle „projekte“ übereinstimmt.

4.2.6 Funktionen der Partner- / Lieferantenverwaltung

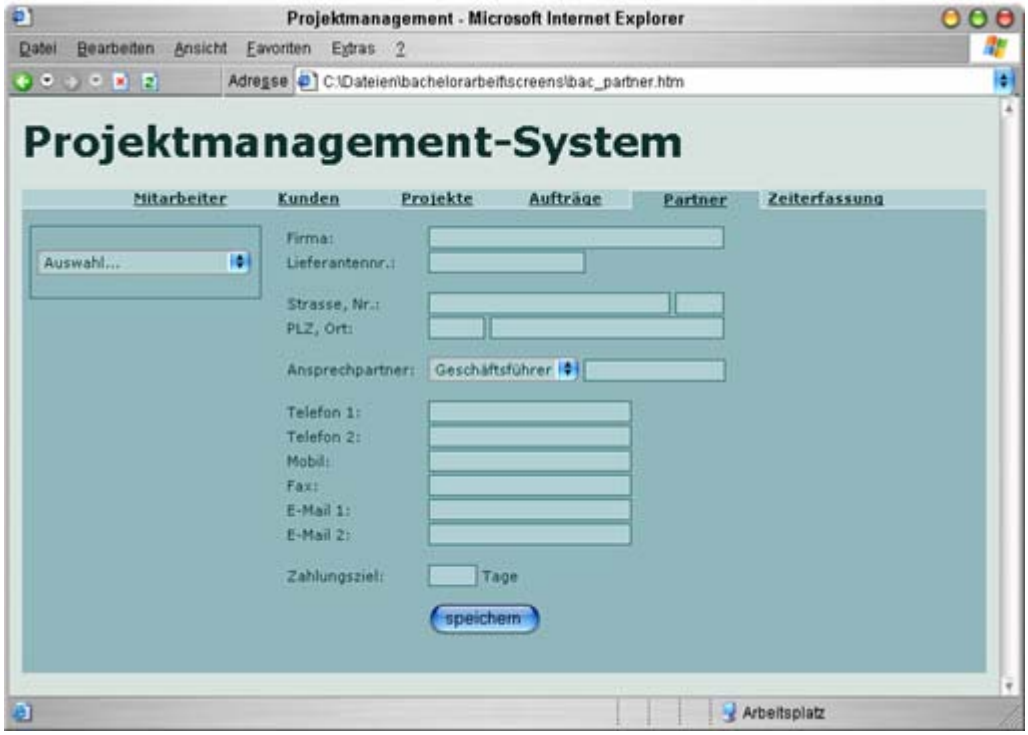


Abbildung 4.2.6.1: Programmoberfläche der Partner-/Lieferantenverwaltung

Abbildung 4.2.6.1 stellt die Partnerverwaltung, die das Management der Daten von externen Partnern oder Lieferanten beinhaltet, dar. In der grafischen Bedienoberfläche ist dies der Karteireiter „Partner“.

Nach Verbindungsaufnahme zum Datenbank-System mittels der PHP-Funktionen mysql_connect() und mysql_select_db() und Auslesen der Tabelle „partner“ („select * from partner where XSTATUS='aktiv'“) werden im linken Drop-Down-Menü alle im System befindlichen Partner/Lieferanten dargestellt, so dass dort ein Partner ausgewählt wird, dessen Daten, nach Extraktion aus der Entität, in den dazugehörigen Textfeldern angezeigt werden.

Weiterhin bietet dieses Menü die Möglichkeit, neue Partner/Lieferanten anzulegen bzw. Daten von existierenden Partnern/Lieferanten zu löschen.

Zu beachten ist hier wieder, dass das Löschen als Deaktivieren des Datensatzes eines Partners/Lieferanten anzusehen ist. Das heißt, das Datenbank-Feld „XSTATUS“ wird auf „inaktiv“ gesetzt. Dies erledigt der SQL-Befehl: „update partner set XSTATUS=’inaktiv’ where ID=’$id_partner’“, wobei „$id_partner“ die ID des zu deaktivierenden Datensatzes eines Partner/Lieferanten ist, der im Drop-Down-Feld ausgewählt wurde.

Eine interne Lieferantennummer spezifiziert einen Partner.

Weiterhin wird ein Zahlungsziel, innerhalb dessen offene Rechnungen zu begleichen sind, im System gespeichert.

Warnhinweise geben Auskunft über fehlerhaft ausgefüllte Textfelder.

4.2.7 Funktionen der Zeiterfassung

Im Bereich der Zeiterfassung trägt ein Mitarbeiter seine zur Ausführung einer bestimmten Tätigkeit bzw. eines bestimmten Jobs benötigte Arbeitszeit und benötigte Ressourcen ein. Ressourcen sind hier als benötigte Materialien oder andere in eine Rechnung einzubeziehende Posten, wie zum Beispiel Fahrtkosten, anzusehen. Abbildung 4.2.7.1 zeigt ein Frontend, wie es zur Zeiterfassung benutzt wird.

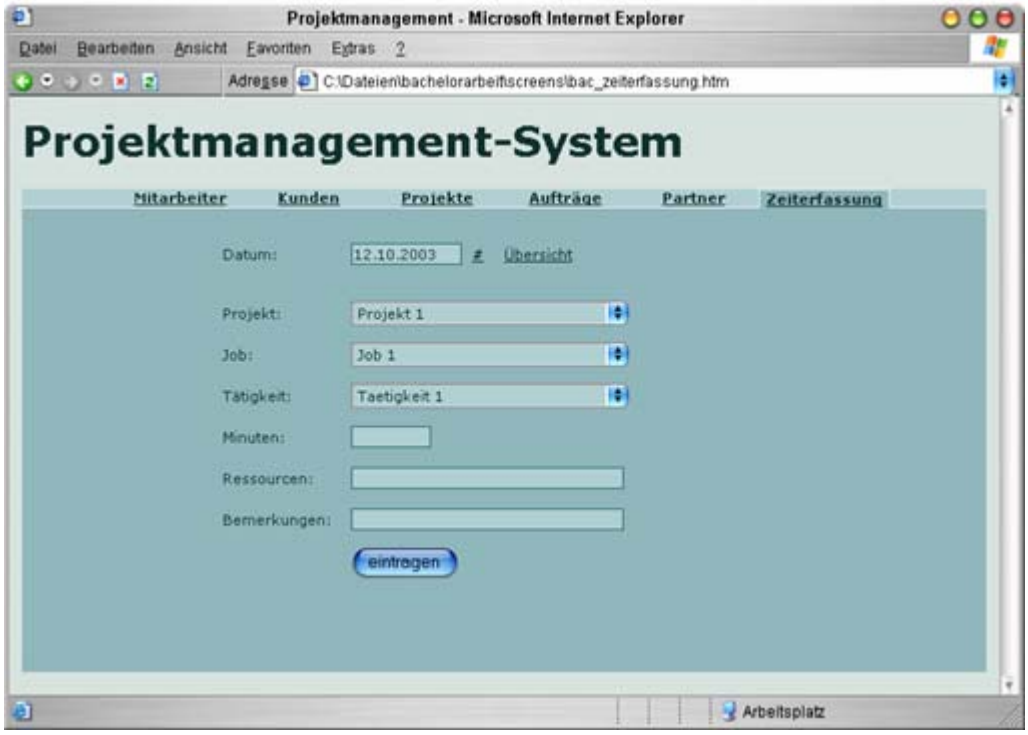


Abbildung 4.2.7.1: Programmoberfläche der Zeiterfassung

Da eine für die Nutzung des Projektmanagement-Systems nötige Anmeldung erfolgt ist, wurde dort beim Start der Nutzungs-Session die ID des Mitarbeiters hinterlegt. Dadurch wird beim Eintragen der Zeiterfassungsdaten in die Datenbanktabelle „zeiterfassung“ die Mitarbeiter-ID als Fremdschlüssel „FID_MITARBEITER“ hinterlegt.

Nach Eingabe des Datums bzw. Auswahl über den Platzhalter „#“ wählt ein Mitarbeiter das Projekt aus. Dieses Drop-Down-Feld wird mit allen im System befindlichen Projekten befüllt, indem die Datenbanktabelle „projekte“ ausgelesen wird („select * from projekte where XSTATUS=’aktiv’“).

Ist ein Projekt ausgewählt, wird das Drop-Down-Feld „Job“ mit den aus der Datenbanktabelle „jobs“ stammenden Datensätzen generiert. Da über den Fremdschlüssel „FID_PROJEKTE“ die Entität „projekte“ referenziert wird, ist somit sicher gestellt, dass nur die zu einem bestimmten Projekt gehörenden Jobs aus der Datenbank extrahiert werden. Der SQL-Query dazu lautet: „select * from jobs where FID_PROJEKTE=’$id_projekte’“, wobei $id_projekte die ID des ausgewählten Projektes ist.

Nachdem wiederum ein Job ausgewählt wurde, erfolgt ein Auslesen aller in der Datenbanktabelle „jobs_taetigkeiten“ befindlichen Datensätze, deren „FID_JOBS“ mit der ID des aktuellen Jobs übereinstimmt („select * from jobs_taetitgkeiten where FID_JOBS=’$arr_jobs[“ID“]’“). Anschließend werden alle Datensätze aus der Entität „taetigkeiten“ extrahiert, deren ID der „FID_TAETIGKEITEN“ aus der Tabelle „jobs_taetigkeiten“ entspricht („select * from taetigkeiten where ID = ’$arr_jobstaetigkeiten[FID_TAETIGKEITEN]’“).

Nachdem in der Programmoberfläche die benötigte Arbeitszeit in das entsprechende Textfeld eingetragen wurde und nötige Bemerkungen und benötigte Ressourcen hinzugefügt wurden, schließt die Schaltfläche „eintragen“ den Vorgang der Zeiterfassung ab. Dazu wird mit Hilfe der PHP-Funktionen mysql_connect() und mysql_select_db() die Datenbank konnektiert und in der Tabelle „zeiterfassung“ ein neuer Datensatz hinzugefügt.

Der Fremdschlüssel „FID_MITARBEITER“ ist, wie erwähnt, in der Session gespeichert, so dass eine Identifizierung des Mitarbeiters möglich ist. Um den bearbeiteten Job bzw. die bearbeitete Tätigkeit in der Zeiterfassung aufzunehmen, wird die ID des Datensatzes der Tabelle „jobs_taetigkeiten“ als Fremdschlüssel „FID_JOBSTAETIGKEITEN“ in die Entität „zeiterfassung“ gespeichert. Der SQL-Query zur Speicherung der Zeiterfassungsdaten lautet: „insert into zeiterfassung (FID_MITARBEITER, FID_JOBSTAETIGKEITEN, DATUM, MINUTEN, RESSOURCEN, BEMERKUNGEN, EINTRAGSDATUM, XSTATUS) values (’$id_mitarbeiter’, ’$id_jobstaetigkeiten’, ’$datum’, ’$minuten’, ’$ressourcen’, ’$bemerkungen’, ’$eintragsdatum’, ’aktiv’)“, wobei die „$id_mitarbeiter“ die aktuelle ID des angemeldeten Mitarbeiters und die „$id_jobstaetigkeiten“ die ID des Datensatzes aus der Datenbank-Tabelle „jobs_taetigkeiten“ ist. Das Datum wird als Timestamp in der Datenbank abgelegt. Dazu erzeugt die PHP-Funktion „timestamp()“, in deren Argumentenliste der Tag, Monat und das Jahr angegeben werden, den zugehörigen Zeitstempel. Die gleiche PHP-Funktion kommt noch einmal beim Eintragsdatum zur Anwendung, nur dass dort keine Argumente übergeben werden, so dass der Timestamp aus der aktuellen Systemzeit generiert wird. Die Inhalte der Variablen $minuten, $ressourcen und $bemerkungen werden nach einer Plausibilitätsüberprüfung so in die Datenbank abgespeichert.

5. Vergleich zu anderen am Markt befindlichen Projektmanagement-Systemen

In der Abbildung 5.1 ist eine Tabelle dargestellt, die einen kurzen Überblick der Eigenschaften verschiedener Projektmanagement-Systeme bietet.
Aus der Vielzahl der am Markt befindlichen Systeme ist nur eine Auswahl von drei weiteren Projektmanagement-Tools, die als Vergleich zu dem in dieser Arbeit beschriebenen System (PMS) dienen, aufgelistet.

	PMS	**MS Project 2002**	**fx-project**	**Genius Enterprise Projects**
Architektur	web-basiert	Standalone-System	web-basiert	web-basiert, Lotus Domino
DB-System	MySQL	integriertes Dateiformat, ODBC	MS-SQL, PostgreSQL, ODBC	ODBC
Sprache	deutsch	deutsch	deutsch, englisch	englisch

Abbildung 5.1: Vergleich verschiedener Projektmanagement-Systeme

Wie in Abbildung 5.1 dargestellt, sind in dieser Tabelle jeweils Vertreter von Projektmanagement-Systemen, die unter verschiedenen Systemarchitekturen betrieben werden, dargestellt.

Die web-basierten Systeme benötigen zum Betrieb einen eigenständigen Server, inkl. eines Datenbank-Systems. Verschiedene Clients greifen über grafische Benutzeroberflächen, meist über Browser, auf den gemeinsamen Serverprozess zu, um die Funktionalitäten eines solchen Projektmanagement-Systems zu nutzen. Ein klassisches Beispiel ist hierfür „fx-project“ der FeRox Management Consulting GmbH.

Dabei ist es auch möglich, auf vorhandene Serverlösungen zurückzugreifen, die das Projektmanagement-System einbeziehen und so das gesamte System erweitern. Ein Beispiel dafür ist das System „Genius Enterprise Projects“ der Genius Inside Germany. Dieses setzt als zu Grunde liegende Systemarchitektur einen Lotus Domino Server voraus, der als Groupware-Server funktioniert. Ein Vorteil einer solchen Lösung ist die optimale Zusammenarbeit der verschiedenen Server-Teilsysteme. Beispielsweise stellt das Lotus Domino System vorhandene Mitarbeiterdaten sofort dem Projektmanagement-System zur Verfügung. Auch ist es möglich, das vorhandene Messaging-System der Groupware-Lösung in das Projektmanagement-System zu integrieren, um projektbezogene Termine zu planen und diese den Mitarbeitern zur Verfügung zu stellen.

Einen andere Arbeitsweise verfolgt Microsoft Project 2002. Dieses als Standalone-System betriebenes Projektmanagement-System beinhaltet alle für den Betrieb nötigen Ressourcen in einer einzigen Applikation. Auch ist es hier möglich, auf einen gemeinsamen Microsoft Project Server zuzugreifen und die Projektdaten der verschiedenen Mitarbeiter abzugleichen. Voraussetzung dafür aber ist der Betrieb des kompletten Projektmanagement-Systems auf jedem Client.

6. Zusammenfassung

In dieser Arbeit wurden die theoretischen Grundlagen, die zur Implementierung eines Projektmanagement-System benötigt werden, behandelt.
In einer Einleitung sind die Gründe, die zu einer Erstellung eines solchen Systems führen, diskutiert worden. Dabei stellte sich heraus, dass es von Vorteil ist, im Umfeld einer Full-Service-Media-Agentur ein auf deren Erfordernisse ausgerichtetes Projektmanagement-System neu zu implementieren. Dabei ist auch ein marktrelevanter Aspekt nicht zu vernachlässigen, da sich eine Veräußerung des Systems an Kunden anbietet.

Weiterhin wurden die allgemeinen Anforderungen und konzeptionellen Vorstellungen für einen Aufbau eines Projektmanagement-Systems behandelt. Dazu gehören Diskussionen über die allgemeinen Funktionalitäten, d. h. wie das zukünftige Projektmanagement-System in seiner Gesamtheit funktioniert. Außerdem sind Anforderungen der einzelnen Themenbereiche des Systems - die Mitarbeiterverwaltung, die Kundenverwaltung, die Projektverwaltung, die Auftragsverwaltung, die Partner-/Lieferantenverwaltung und der Themenbereich der Zeiterfassung - erörtert worden.

Ferner sind in einem Abschnitt Aspekte zur Auswahl der Systemarchitekur betrachtet worden. Dazu erfolgte, nach grundlegender Diskussion über Vor- und Nachteile diverser Betriebsarten, ein Performance-Vergleich von verschiedenartigen Programmierarten.

Im Ergebnis dieses Vergleiches ist es empfehlenswert, das System als PHP-Server-Applikation unter einem HTTP-Server-System zu betreiben.

Anschließend wurde die Datenbankstruktur und deren einzelne Entitäten besprochen. Auf Besonderheiten diverser Datenbank-Elemente und deren Interaktion mit anderen Entitäten der Datenbank wurde im Einzelnen eingegangen.

Es wurden ausführlich die Funktionen der einzelnen Themenebereiche des Projektmanagement-Systems dargelegt. Unterstützend wirkten grafische Benutzeroberflächen, die die Funktionalitäten des Systems detaillierter darstellten. Anhand der Frontends wurde das Zusammenspiel der Oberfläche mit dem zu Grunde liegenden Server-System erläutert.

Ein Vergleich mit anderen am Markt befindlichen Projektmanagement-Systemen gab einen Überblick über Funktionalitäten anderer Projektmanagement-Tools. Dazu wurden bewusst aus der Vielzahl solcher Systeme 3 unter verschiedenen Betriebsarten arbeitenden Projektmanagement-Systeme als Vergleich herangezogen.

Abschließend erfolgte ein Ausblick auf zukünftige Erweiterungen des Projektmanagement-Systems, wo Verbesserungen und Optimierungen erörtert wurden, um das bestehende System weiterhin auszubauen.

7. Zukünftige Erweiterungen des Projektmanagement-Systems

In dieser Arbeit wurden die grundlegenden theoretischen Spezifikationen zur Implementierung des Projektmanagement-System erörtert, sowie genauere Beschreibungen der Grundfunktionen desselben gegeben.
Natürlich kann und wird ein solches System ständig erweitert und verbessert werden, für eine Marktreife aber reichen die bisherigen Funktionalitäten schon aus.

Erweiterungsmöglichkeiten bestehen zum Beispiel im Bereich des Work-Flow-Management. Dabei helfen erstellte Statistiken über Arbeitsaufwand, Kosten und Nutzen, den Arbeitsablauf zu optimieren. Auch bietet ein Gantt-Plan einen Überblick über aktuelle Projekte.

Auch ist die Integration einer Schnittstelle zu anderen Applikationen denkbar, um zum Beispiel auf Benutzerdaten aus einer zentralen Datenbank zugreifen zu können, die auch anderen Anwendungen zur Verfügung steht, beispielsweise das Windows-eigene Adressbuch oder auch professionelle Kontaktverwaltungs-Programme. Außerdem ist so eine Synchronisation der Termine (Projektstart, Projektabgabe, etc.) mit vorhandenen Terminverwaltungs-Tools, beispielsweise Microsoft Outlook, möglich.

Weiterhin bietet sich eine projektbezogene Mitarbeiterverwaltung an. Mitarbeiter haben nur Zugriff auf bestimmte Projekte, welche ein Projektleiter bzw. ein Systemmanager einer bestimmten Mitarbeitergruppe frei gibt.
Auch gibt es die Möglichkeit, die Ausführung von Jobs an bestimmte Mitarbeiter zu binden. Damit erhält ein Mitarbeiter eine Übersicht über den aktuellen Bearbeitungsstand seiner auszuführenden Jobs.

Als Erweiterung lassen sich den einzelnen Mitarbeitern bestimmte Fähigkeiten zuzuordnen. So erhält ein Projektleiter einen Überblick über Mitarbeiter, die für die Erfordernisse einer bestimmten Tätigkeit besonders geeignet sind. Ein gezielter Einsatz dieser Mitarbeiter beschleunigt somit die Ausführungszeiten von Projekten, da Einarbeitungszeiten sowie Vorbereitungszeiten der Mitarbeiter wegfallen.

Eine virtuelle Kontoführung stellt einen Überblick über vorhandene Ein- und Ausgaben dar. Damit ist beispielsweise eine einfache Liquiditätsvorschau realisierbar.

8. Literaturverzeichnis

[DIN1-03] Deutsches Institut für Normung e. V., Definition „Projektmanagement-System“ nach DIN 69905, Berlin, 2003, www.din.de

[SFTW-01] „Macromedia Homesite 5“, 2001, Macromedia Inc.

[SFTW-03] fabFORCE “DBDESIGNER 4”, Michael G. Zinner, 2003, www.fabforce.net

[SFTW-04] fx-project, FeRox Management Consulting GmbH, Passau, 2003, www.ferox.de

[SFTW-05] Genius Enterprise Project, Genius Inside Germany, Aachen, 2003, www.geniusinside.com

[SFTW-06] Microsoft Project 2002, Microsoft Deutschland GmbH, 2003, www.microsoft.com/germany/

9. Abbildungsverzeichnis

10. Anhang

Quellcode des Servertest-Skriptes „mysql_read.php“

Quellcode des Servertest-Skriptes „mysql_read.cgi“

Quellcode des Servertest-Skriptes „mysql_read.jsp“

Quellcode des Servertest-Skriptes „mysql_read.aspx“

Datenbank-Schema des Projektmanagement-Systems

mysql_read.php:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">

<html>
<head>
	<title></title>
	<style>
		body{
			font-family:verdana;
			font-size:8pt;
		}
	</style>
</head>

<body>
<?php
//Start-Zeit in Variable '$start' speichern [sec]
$start=microtime();

//Datenbankverbindung herstellen
$conn=mysql_connect("127.0.0.1","root","root");

//Datenbank auswählen
mysql_select_db("bac");

//SQL-Query erzeugen und Datensätze in Array '$query_read' speichern
$query_read=mysql_query("select * from bac order by FELD1");

//Auslesen des Datensatz-Arrays '$query_read' und Ausgabe dessen
while($arr_read=mysql_fetch_array($query_read)){
	echo "FELD1: ".$arr_read["FELD1"]."<br>FELD2:
	".$arr_read["FELD2"]."<br>FELD3: ".$arr_read["FELD3"]."<br>FELD4:
	".$arr_read["FELD4"]."<br><br>";
}

//Datenbankverbindung schließen
mysql_close($conn);

//Ende-Zeit in Variable '$ende' speichern [sec]
$ende=microtime();

//Zeit-Differenz bilden und in Variable '$zeit' speichern
$zeit=($ende-$start)*1000;

//Ausgabe der Zeitdifferenz
echo "<hr>".$zeit." ms";
?>


</body>
</html>
```

mysql_read.cgi:

```
#!c:/perl/bin/perl.exe
print "Content-type: text/html\n\n";

# Modul für genaue Zeitmessung laden
use Time::HiRes qw(gettimeofday tv_interval);

# Start-Zeit in Variable 'tstart' speichern
$tstart=[gettimeofday];

# Modul für Datenbank-Verbindungen laden
use DBI;

# Variablendefinitionen: DB-Treiber, DB-Name, DB-Host, DB-User, DB-Passwort
my $driver="mysql";
my $database="bac";
my $hostname="127.0.0.1";
my $user="root";
my $password="root";

# MySQL-DB-Verbindungs-String mittels definierten Variablen erzeugen
my $dsn = "DBI:$driver:database=$database;host=$hostname;";

# Verbindung zur Datenbank herstellen
my $dbh=DBI->connect("DBI:$driver:$dsn",$user,$password)||die "Keine
        Verbindung zum MySQL-Server: $DBI::errstr\n";

# SQL-Query erzeugen
my $statement="SELECT * FROM bac order by FELD1";

# SQL-Query vorbereiten
my $sth=$dbh->prepare($statement)||die $dbh->errstr;

# SQL-Query ausführen und in Array '$sth' speichern
my $success=$sth->execute()||die $sth->errstr;

#Auslesen des Datensatz-Arrays '$sth' und Ausgabe dessen
while(my ($id,$field1,$field2,$field3,$field4) = $sth->fetchrow){
        print
        "FELD1:$field1<br>FELD2:$field2<br>FELD3:$field3<br>FELD4:$field4
        <br><br>";
}
#Datensatz-Array schließen
$sth->finish;

#Datenbank-Verbindung schließen
$dbh->disconnect;

#Zeitdifferenz zwischen Start-Zeit und aktueller Zeit berechnen und in Variable
        '$time' speichern
my $time=tv_interval($tstart, [gettimeofday])*1000;

#Ausgabe der Zeitdifferenz
print "<hr>".$time."<br>";
```

mysql_read.jsp:

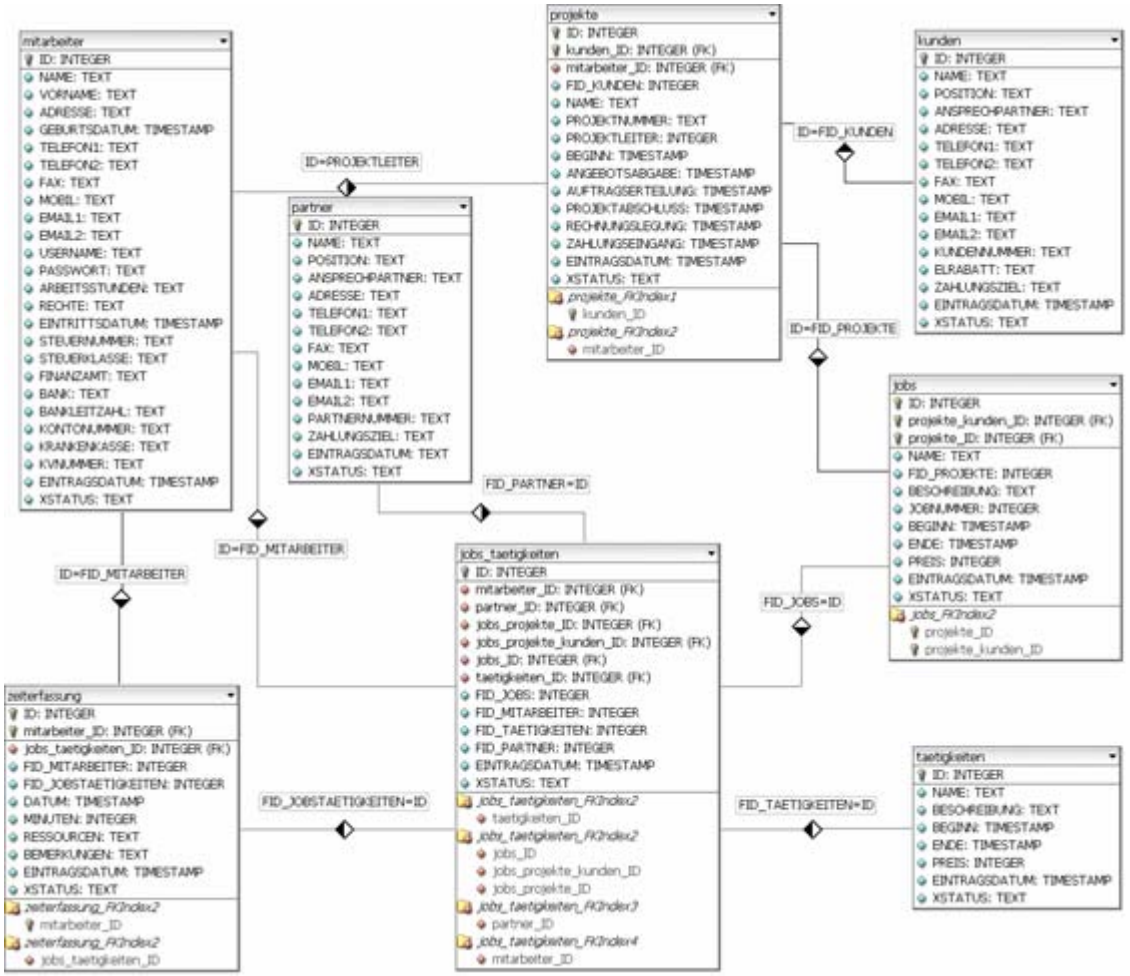
```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">

<html>
<head>
	<title></title>
	<style>
		body{
			font-family:verdana;
			font-size:8pt;
		}
	</style>
</head>

<body>
<!--Java-Package importieren-->
<%@ page language="java" import="java.sql.*" %>
<%
//Start-Zeit in Variable 'start' speichern [ms]
long start=System.currentTimeMillis();

//Variablendefinition
long ende;
long zeit;
String host="127.0.0.1";
String user="root";
String pass="root";
Statement stmt=null;
ResultSet rs=null;

//MySQL-Verbindungsklasse initialisieren
try{
	Class.forName("com.mysql.jdbc.Driver");

//Datenbankverbindung herstellen
	try{
		Connection conn =
	DriverManager.getConnection("jdbc:mysql://192.168.1.1/bac?user=roo
	t");

//SQL-Query erzeugen und Datensätze in ResultSet 'rs' speichern
		stmt=conn.createStatement();
		rs=stmt.executeQuery("select * from bac order by FELD1");

//Auslesen des ResultSets 'rs' und ausgeben
		while(rs.next()){
			out.print("FELD1: "+rs.getString(2)+"<br>");
			out.print("FELD2: "+rs.getString(3)+"<br>");
			out.print("FELD3: "+rs.getString(4)+"<br>");
			out.print("FELD4: "+rs.getString(5)+"<br><br>");
		}

//Ende-Zeit in Variable 'ende' speichern
		ende=System.currentTimeMillis();
```

```
//Zeit-Differenz bilden und in Variable 'zeit' speichern
		zeit=ende-start;

//Ausgabe der Zeit-Differenz
		out.println("<hr>");
		out.println(zeit+" ms");

//Abfangen der SQL-Fehlermeldungen
	}catch(SQLException sql_e){
		out.println("Datenbankverbindung konnte nicht hergestellt werden
	!!!");
	}

//Abfangen der SQL-Klassen-Fehlermeldungen
}catch(ClassNotFoundException cnf_e){
	out.println("Klasse der Datenbankverbindung konnte nicht geladen
	werden !!!");
}
%>


</body>
</html>
```

mysql_read.aspx:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">

<html>
<head>
        <title></title>
        <style>
            body{
                font-family:verdana;
                font-size:8pt;
            }
        </style>
</head>

<body>
<%@ Page aspcompat=true debug=true%>

<%
Dim conn
Dim rs
Dim dbstring
Dim command
Dim start As DateTime
Dim ende As DateTime
Dim zeit
start=now

dbstring = "DRIVER=MySQL ODBC 3.51
        Driver;"&"SERVER=127.0.0.1;HOST=127.0.0.1;"&"DATABASE=bac;"&"
        UID=root;"&"PWD=root;"

'// Datenbank-Verbindung referenzieren und öffnen
conn = Server.CreateObject("ADODB.Connection")
conn.Open(dbstring)

'// SQL-Query erzeugen und absetzen
command = "select * from bac order by FELD1"
rs = conn.Execute(command)

While Not rs.EOF
   Response.Write("FELD1: ")
   Response.Write(rs("FELD1"))
   Response.Write("<br>")
   Response.Write("FELD2: ")
   Response.Write(rs("FELD2").toString)
   Response.Write("<br>")
   Response.Write("FELD3: ")
   Response.Write(rs("FELD3"))
   Response.Write("<br>")
   Response.Write("FELD4: ")
   Response.Write(rs("FELD4"))
   Response.Write("<br>")
   rs.MoveNext()
End While
```

```
rs.Close()
conn.Close()
rs = Nothing
conn = Nothing

ende=now
zeit=(ende.Millisecond+ende.Second*1000+ende.Minute*60000)-
        (start.Millisecond+start.Second*1000+start.Minute*60000)
Response.Write("<hr>Zeit: "&zeit)
%>

</body>
</html>
```


mitarbeiter
ID: INTEGER
NAME: TEXT
VORNAME: TEXT
ADRESSE: TEXT
GEBURTSDATUM: TIMESTAMP
TELEFON1: TEXT
TELEFON2: TEXT
FAX: TEXT
MOBIL: TEXT
EMAIL1: TEXT
EMAIL2: TEXT
USERNAME: TEXT
PASSWORT: TEXT
ARBEITSSTUNDEN: TEXT
RECHTE: TEXT
EINTRITTSDATUM: TIMESTAMP
STEUERNUMMER: TEXT
STEUERKLASSE: TEXT
FINANZAMT: TEXT
BANK: TEXT
BANKLEITZAHL: TEXT
KONTONUMMER: TEXT
KRANKENKASSE: TEXT
KVNUMMER: TEXT
EINTRAGSDATUM: TIMESTAMP
XSTATUS: TEXT
projekte
ID: INTEGER
kunden_ID: INTEGER (FK)
mitarbeiter_ID: INTEGER (FK)
FID_KUNDEN: INTEGER
NAME: TEXT
PROJEKTNUMMER: TEXT
PROJEKTLEITER: INTEGER
BEGINN: TIMESTAMP
ANGEBOTSABGABE: TIMESTAMP
AUFTRAGSERTEILUNG: TIMESTAMP
PROJEKTABSCHLUSS: TIMESTAMP
RECHNUNGSLEGUNG: TIMESTAMP
ZAHLUNGSEINGANG: TIMESTAMP
EINTRAGSDATUM: TIMESTAMP
XSTATUS: TEXT
projekte_FKIndex1
kunden_ID
projekte_FKIndex2
mitarbeiter_ID
kunden
ID: INTEGER
NAME: TEXT
POSITION: TEXT
ANSPRECHPARTNER: TEXT
ADRESSE: TEXT
TELEFON1: TEXT
TELEFON2: TEXT
FAX: TEXT
MOBIL: TEXT
EMAIL1: TEXT
EMAIL2: TEXT
KUNDENNUMMER: TEXT
ELRABATT: TEXT
ZAHLUNGSZIEL: TEXT
EINTRAGSDATUM: TIMESTAMP
XSTATUS: TEXT
partner
ID: INTEGER
NAME: TEXT
POSITION: TEXT
ANSPRECHPARTNER: TEXT
ADRESSE: TEXT
TELEFON1: TEXT
TELEFON2: TEXT
FAX: TEXT
MOBIL: TEXT
EMAIL1: TEXT
EMAIL2: TEXT
PARTNERNUMMER: TEXT
ZAHLUNGSZIEL: TEXT
EINTRAGSDATUM: TEXT
XSTATUS: TEXT
jobs
ID: INTEGER
projekte_kunden_ID: INTEGER (FK)
projekte_ID: INTEGER (FK)
NAME: TEXT
FID_PROJEKTE: INTEGER
BESCHREIBUNG: TEXT
JOBNUMMER: INTEGER
BEGINN: TIMESTAMP
ENDE: TIMESTAMP
PREIS: INTEGER
EINTRAGSDATUM: TIMESTAMP
XSTATUS: TEXT
jobs_FKIndex2
projekte_ID
projekte_kunden_ID
jobs_taetigkeiten
ID: INTEGER
mitarbeiter_ID: INTEGER (FK)
partner_ID: INTEGER (FK)
jobs_projekte_ID: INTEGER (FK)
jobs_projekte_kunden_ID: INTEGER (FK)
jobs_ID: INTEGER (FK)
taetigkeiten_ID: INTEGER (FK)
FID_JOBS: INTEGER
FID_MITARBEITER: INTEGER
FID_TAETIGKEITEN: INTEGER
FID_PARTNER: INTEGER
EINTRAGSDATUM: TIMESTAMP
XSTATUS: TEXT
jobs_taetigkeiten_FKIndex2
taetigkeiten_ID
jobs_taetigkeiten_FKIndex2
jobs_ID
jobs_projekte_kunden_ID
jobs_projekte_ID
jobs_taetigkeiten_FKIndex3
partner_ID
jobs_taetigkeiten_FKIndex4
mitarbeiter_ID
zeiterfassung
ID: INTEGER
mitarbeiter_ID: INTEGER (FK)
jobs_taetigkeiten_ID: INTEGER (FK)
FID_MITARBEITER: INTEGER
FID_JOBSTAETIGKEITEN: INTEGER
DATUM: TIMESTAMP
MINUTEN: INTEGER
RESSOURCEN: TEXT
BEMERKUNGEN: TEXT
EINTRAGSDATUM: TIMESTAMP
XSTATUS: TEXT
zeiterfassung_FKIndex2
mitarbeiter_ID
zeiterfassung_FKIndex2
jobs_taetigkeiten_ID
taetigkeiten
ID: INTEGER
NAME: TEXT
BESCHREIBUNG: TEXT
BEGINN: TIMESTAMP
ENDE: TIMESTAMP
PREIS: INTEGER
EINTRAGSDATUM: TIMESTAMP
XSTATUS: TEXT
ID=PROJEKTLEITER
ID=FID_KUNDEN
ID=FID_PROJEKTE
ID=FID_MITARBEITER
ID=FID_MITARBEITER
FID_PARTNER=ID
FID_JOBS=ID
FID_JOBSTAETIGKEITEN=ID
FID_TAETIGKEITEN=ID