



BEUTH HOCHSCHULE FÜR TECHNIK BERLIN

University of Applied Sciences



Beuth Hochschule für Technik Berlin
Fachbereich VI - Informatik und Medien
Luxemburger Str. 10, 13353 Berlin

Bachelorarbeit

Entwicklung einer Datenbank-Applikation für das "acaLoan-Raspi"-Projekt des PSE-Labors

**Development of a database application for the "acaLoan-Raspi"
project of the PSE laboratory**

Oleksandra Baga

Beuth Hochschule für Technik Berlin
Matrikelnummer 849852
Bachelor of Engineering (B.Eng.)
Computer Engineering - Embedded Systems
E-Mail: oleksandra.baga@gmail.com

Betreuer Prof. Dr. Christian Forler
Fachbereich VI - Informatik und Medien
Beuth Hochschule für Technik Berlin

Gutachter Prof. Dr. Görlich
Fachbereich VI - Informatik und Medien
Beuth Hochschule für Technik Berlin

31.10.2020

Contents

Abbildungsverzeichnis	1
Abkürzungsverzeichnis	2
1 Einleitung	4
1.1 Motivation und Aufgabestellung	4
1.2 Aufbau der Arbeit	5
2 Theoretische Grundlagen	8
2.1 Grundlagen der Webanwendungen	9
2.2 Raspberry Pi Board und Betriebssystem	12
2.3 Kontaktlose Smartcards MIFARE	15
2.4 Sender-Empfänger-System mit RFID	17
2.5 HTTP für Design der verteilten Systeme	20
2.6 Django Framework	20
2.7 Grundlagen der REST API	22
2.8 Clientseitiges JavaScript	22
3 Systemdesign	23
3.1 User Stories	23
3.2 Anforderungen	24
3.2.1 Funktionale Anforderungen	24
3.2.2 Nichtfunktionale Anforderungen	25
3.3 Systemmodell mit UML	26
3.4 Komponentendiagramm	26
3.4.1 Klassendiagramm	26
3.4.2 Anwendungsfalldiagramm	26
3.5 Systemarchitektur	27
3.6 Endliche Zustandsmaschine	28
4 Implementierung	29
4.1 Register-Client	29
4.1.1 Installation des Betriebssystem	30
4.1.2 Installation der Treibers für RFID-Leser	31
4.1.3 Spannungsproblem und Lösung	33

4.1.4	Python Programmierung des RFID-Lesers	33
4.2	Server	37
4.2.1	Erstellung acaLoan Django-Projekts	38
4.2.2	Datenbankeinrichtung	40
4.2.3	Erstellen der öffentlichen Schnittstelle - "Views"	40
4.2.4	Einführung in das Django Admin	40
4.2.5	Templates und Design der acaLoan-Website	40
4.2.6	Implementierung des Use Cases	40
4.3	Display-Client	41
4.3.1	Clientseitiges JavaScript	41
4.4	Automatisierten Tests	43
5	Results and conclusion	44
5.1	Results	44
5.2	Conclusion	44
Glossar		45
Literaturverzeichnis		50

Abbildungsverzeichnis

Abb. 1	Das Raspi Loan-Board	7
Abb. 2	Schematische Darstellung der Client-Server Kommunikation	11
Abb. 3	Peripherieanschlüsse des Mikrocomputers	12
Abb. 4	The Raspberry Pi's system-on-chip (SoC)	13
Abb. 5	Mifareproduktfamilie	16
Abb. 6	Verschiedenen RFID-Tags	18
Abb. 7	UML Klassendiagramm.	27
Abb. 8	ACR122U-A9 von Advanced Card Systems und Raspberry Pi	32
Abb. 9	RFID-Leser mit der Studentenkarte der Autorin und RFID-Transponders	35
Abb. 10	Screenshot von Raspberry Pi nach dem Erstellung des acaLoanRaspiBoard-Projekts.	39
Abb. 11	Sequenzdiagramme der erfolgreichen Ausleihe der Lab-Loan Raspi Board	42

Abkürzungsverzeichnis

AJAX Asynchrones Javascript und XML

API Application Programming Interface

ASGI Asynchronous Server Gateway Interface

ATR Answer-To-Reset

BGA Ball Grid Array

CPU Central processing unit

DOM Document Object Model

GPU Graphic processing unit

HTTP HyperText Transfer Protocol

JSON JavaScript Object Notation

IDE Integrated Development Environment

IEEE Institute of Electrical and Electronics Engineers

ISO International Organization for Standardization

MAC Media Accesss Control Layer

NFC Near Field Communication

SDRAM Synchronous dynamic random-access memory

SoC System-on-Chip

RAM Random-access memory

Raspi Raspberry Pi Board und Minicomputer im PSE-Labor der BHS

RFID Radio-frequency identification

URI Uniform Resource Identifier, Unified Resource Identifier

URN Uniform Ressource Name

Einleitung

1.1 Motivation und Aufgabestellung

Die vorliegende Arbeit beschäftigt sich mit der Entwicklung einer Datenbank-Applikation für das PSE-Labor¹ (Labor für Pervasive Systems Engineering), das sich an der Beuth Hochschule für Technik Berlin befindet und seit fast 5 Jahre ein wichtiger Teil des Studiums im Studiengänge Technische Informatik und Medieninformatik ist. Die zwei Mitarbeiter, Andreas Döpkens und Brian Schüler, beitragen selbstmotiviert zur den Projekte, die zwar räumlich im PSE-Labor der Beuth-Hochschule sind und betrieben werden können², wurden aber als "das acaLab" genannt, bei dem es sich sozusagen um ein virtuelles Labor in einem reellen Labor handelt. Die acaLab-Projekte sollen als interessante Mitarbeiter-Beiträge neben den individuellen Tätigkeiten im Fachbereich 6 der Beuth-Hochschule bereichern und werden deshalb auch exemplarisch jedes Jahr auf der Langen Nacht der Wissenschaften³ gezeigt. An dieser Stelle ist auch noch anzumerken, die acaLab-Projekte aus den Mitteln des PSE-Labors finanziert werden und damit möglichst viele Studierende für lehr- und erkenntnisreichen Abschlussarbeiten eingeladen sind, de der Vergabe die Aufgaben aus den acaLab-Projekten als Abschlussarbeiten spart dem Fachbereich ein Budget. Die Erkenntnisse und Ergebnisse aus den acaProjekt-Tätigkeiten sind später in die Lehre einfließen zu lassen.[TBa] Meine Abschlussarbeit widmet sich dem neuen Projekt des Labors namens "acaLoan-Raspi".

Während im PSE-Labor stattfindenden Übungsveranstaltungen im Studiengang Technische Informatik werden vorhandene im PSE-Labor die Raspberry Pi Minicomputers (kurz: Raspi) an die Studierenden verliehen. Zu Beginn einer Laborübung werden die Raspi Boards den Studierenden vom Lehrkraft, der die Übung betreut, übergeben und am Ende der Laborübung zurückgezogen. Aus 16 vorhandenen im Labor Raspis, die markierte mit den Nummern 12-16 von Studierenden nach Hause (home-loan) genommen werden können.[TBa]

Nachweislich ist das Vorgehen oft mit Reihe von Problemen verknüpft, die sich jedes Semester und fast jedes Mal wiederholen. Die folgenden Problemen wurden

¹<https://www.http://labor.beuth-hochschule.de/pse>

²<https://labor.beuth-hochschule.de/pse/acalab-aktivitaeten-und-abschlussarbeiten/>

³<https://www.langenachtderwissenschaften.de/>

von Mitarbeitern des Labors bereits festgestellt und regelmäßig verlangsamen den Prozess der Verleihung und Übungsführung:

- Studierende kennen ihre am Semesteranfang zugewiesene Gruppennummer auch nach mehreren Wochen nicht und geben den Lehrkraft einen Board mit einer falschen Registriernummer, der einer anderen Gruppe früher zugewiesen wurde und nur von der zugewiesenen Gruppe benutzt werden darf.
- Studierende versuchen einen Board nach Hause auszuleihen, der zu den Lab-Boards gehört und nur im Labor während der Übungszeit verliefert werden darf. Außerplanmäßig von Studierenden darf Lab-Board nicht ausgeliehen und auch mit nach Hause (home-loan) nicht genommen werden.
- Es gibt ein Verwaltungsaufwand für die ausleihbaren Home-Boards, die von den Studierenden für jeweils eine Woche mit nach Hause genommen werden können. Die Mitarbeiter müssen handlich die Studentename, Matrikelnummer, Board und Zeit am Zettel registrieren und in einer Woche überprüfen, ob alle ausgeliehenen Boards pünktlich ins Labor zurückgekommen sind.
- Erfahrungsgemäß können Studierende nach Ablauf der Frist ein Ausleihgerät in einem sehr übeln Zustand der Verschmutzung oder Zerstörung zurückgeben, dass es besteht eine Notwendigkeit den Zustand des Gerätes stets zu kontrollieren, damit es immer bekannt wird, zum welchen Zeitraum Raspi Board zum letzten Mal funktionsfähig war und von wem ausgeliehen wurde.
- Falls gilt ein Raspi Board als verloren, es sollte eine Möglichkeit geben, alle vorherigen Ausleihen anzuschauen und festzustellen, von welchem Studierende es ausgeliehen und nicht zurückgegeben wurde. Mit den Zettelchen, auf denen einen Name von Studierende und eine Board Nummer gemerkt werden, ist es zu aufwändig nachvollziehen.

Somit ist schlusszufolgern, dass eine Notwendigkeit das Verleihprozedere für die Loan-Boards (Lab und Home) mit modernen Mitteln der Technischen Informatik zu lösen schon dringend besteht und eine lohnenswerte Aufgabe für zukünftige Abschlussarbeit ist.

1.2 Aufbau der Arbeit

Obwohl das Thema der Bachelorarbeit "Entwicklung einer Datenbank-Applikation" lautet, ist es keine einzige Aufgabe nur eine Datenbank zur Ausleihverwaltung zu

schrieben ist. Es lässt sich die folgenden Struktur definieren, indem drei verteilte Teilen zu entwickeln ist.

Erstens wird an einem uComputer ein sogenannten Register-Client realisiert. Dafür ein RFID-Leser an uComputer angeschlossen wird. Register-Client ist neben dem Eingangstür eines kleinen Lagerraums des PSE-Labors zu platzieren ist, wo Raspi-Boards aufbewahrt werden. Es ist geplant, dass Studierende einen Board selbst aus dem Fach nehmen könnte und dann mit Hilfe des Register-Clients den genommenen Board auf sich oder seine Gruppe registrieren lassen. Der Register-Client hat selbst keinen Zugriff zur Datenbank und sollte nur die abgelesene Daten von der Smartcard der Studierende zur Server schicken.

Zweitens ist ein Display-Client zu entwickeln, der den Studierenden es zulässt, die Begrüßung des System und eine Beschreibung die für Ausleihe notwendigen Schritte zu sehen. Es sollte in einem Browser-Fenster die aktuelle Server-Kommunikation und Auskunft angezeigt wird, ob die Ausleihe gelang oder ein Fehler aufgetreten war. Ein Android/iOS-Tablett ist eine gute Wahl für die technische Realisierung, da es die Kommunikation zwischen den Mensch und das System leicht und ohne erweiterte Hinweise zulässt.

Drittens ist ein Web-Server für die Datenbank-Applikation schließlich zu implementieren. Er umfasst alle Datensätze über die vorhandenen im Labor Raspi-Boards, registrierten zum Kurs Studierenden und die abgewickelten Leihvorgänge. Web-Server wird mit einem Web-Framework Django erstellt. Django verfügt nun über die Funktionalität und Datenbasis, um die dafür erforderlichen Aktionen durchzuführen. Als Web-Framework bietet Django eine Reihe von Komponenten und Funktionen (Benutzerauthentifizierung, Hochladen von Dateien, Umgang mit Daten usw.), die bei jeder Webanwendung benötigt werden. Mit einem Web-Framework muss ein Entwickler keine Zeit damit verschwenden, denselben Code von Grund auf neu jedes Mal zu schreiben.

Wie erfolgt nun die Abwicklung des eigentlichen Leihvorgangs von der Ausleihe bis zur Rückgabe eines Boards? Zuerst wird eine Studentenkarte am Register-Client ablesen und nachdem sollte einen Name und die Anzahl schon ausgeliehenen Boards am Display-Client angezeigt werden. Falls der Studierende zum Kurs zugelassen ist, darf dann ein gewünschten Board am Register-Client abgelesen werden. Es ist möglich, dass zu den schon ausgeliehenen Lab-Board noch zusätzlich einen Home-Board nach Hause mitgenommen wird. Das abgelesene Board ist entweder auszuleihen oder zurückzugeben. Sämtliche im Verleih befindlichen Geräte werden von den Mitarbeitern des Labors regelmäßig nach jeder Übung vor dem nächsten Ausleihevorgang auf Funktionsfähigkeit geprüft. Es kann sein, dass einem Studierenden die Home-Loan-Absicht eines Boards (12-16) verweigert wird, da in der Vergangenheit schon

einmal vom Studierende ein Board in einem inakzeptabel Zustand zurückgeben war und die Ursachen mit den Mitarbeiter des Labor nicht geklären hat. Falls der Studierende, dem Home-Loan verboten wurde, ein Board auszuleihen versucht, wird eine Fehlermeldung auf dem Bildschirm gezeigt und der Leihvorgang mit dem Fehlerzustand terminiert.

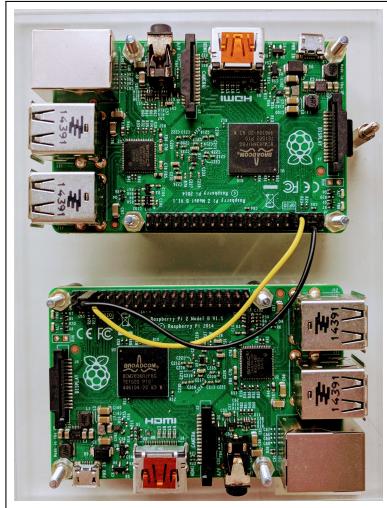


Abb. 1: Das Raspi Loan-Board

Nachdem nun grundlegenden Funktionsweisen der Bestandteile des Verteilten Systems geklärt sind, geht es zur Auswahl der Hardware. Der wichtigste Aspekt beim Hardwarekauf ist einerseits das vorhandene Budget des PSE-Labors und zum Anderen die Aufgaben, die zusammen spielenden Hardware erfüllen sollen. Zunächst müssen die 16 für die Ausleihe zur Verfügung stehenden Raspi Loan-Boards mit einem geeigneten RFID-Tag ausgestattet werden. Es ist selbstklebende NFC Tags zu verwenden, die klein und dünn sind und lassen sich auf der Rückseite des Schutzschirms zu befestigen und ein Aussicht und Funktionen des Geräts nicht zu beeinflussen.

Es ist nicht unerwähnt zu lassen, dass Tags nicht

ausgelesen werden können, wenn diese auf einen metallischen Gegenstand/Oberfläche geklebt wurden, da die Kommunikation aus physischen Gründen zwischen Tag und Lesegerät gestört werden kann. Die Raspi Loan-Boards sind mit einem Schutzschirm aus dem ein transparenter thermoplastischer Kunststoff geschützt und es wurde ins Labor getestet, dass die geklebte auf der Rückseite RFID-Tag lassen sich ablesen und den Board auf RFID Chip-Kartenleser zu platzieren. Erfolgreich getestet wurde RFID Leser/Schreiber ACR122U⁴, der wurde auf der Basis der 13,56 MHz kontaktlosen (RFID) Technologie entwickelt. Der ACR122U USB Kartenleser unterstützt nicht nur Mifare ® Technologien, sondern auch alle vier Typen von NFC -Tags. Diese Anforderung ist für vorliegende Abschlussarbeit wichtig, da Studierenden-Ausweise der Beuth Hochschule mit dieser Technologie gelesen werden können. Des Weiteren soll an dem oben genannten Register-Client der dort angeschlossene RFID Chip-Kartenleser über geeignete Software in Betrieb genommen und die eingelesenen Daten an den Server übermittelt werden. Dafür wird ein Raspberry Pi⁵ verwendet, der zählt zu den beliebtesten Single Board Computern zählt und lässt die anderen Schnittstellen anzubinden. Der kompakte Einplatinencomputer wenig verbraucht Strom aber dennoch genügend Power für ein Linuxbetriebssystem liefert. Detaillierte Beschreibungen zu jeder getroffenen Auswahl finden Sie im entsprechenden Abschnitt des Kapitels "Theoretische Grundlagen"

⁴<https://www.conrad.de/de/p/rfid-reader-acr122u-nfc-usb-802554010.html>

⁵<https://www.raspberrypi.org/>

Theoretische Grundlagen

Als schon im Abschnitt "Einleitung" erwähnt wurde, bei der Aufgabestellung es um eine Entwicklung einer Webanwendung geht. Die zu realisierende Software basiert sich, wie die meisten Webanwendungen, auf einer Client-Server-Architektur, wobei der Client Informationen eingibt, während der Server die eingegebene Daten empfängt, bearbeitet und speichert. Eine Webanwendung ist ein Computerprogramm, das eine bestimmte Funktion unter Verwendung eines Webbrowsers als Client ausführt. Die Webanwendung kann so einfach wie ein Kontaktformular auf einer Website oder so komplex wie eine Textverarbeitungs- oder Bildbearbeitungsprogramm sein, die Sie auf Ihr Computer im Browser ausführen können. Um die Webanwendung zu starten, muss der Benutzer keine zusätzlichen Programme installieren. Sie wird auf jedem Gerät mit Browser und Internetzugang ausgeführt. Der Client ist nicht vom Betriebssystem auf dem Computer des Benutzers abhängig. Bei der Entwicklung von Webanwendungen müssen daher keine separaten Versionen für Windows, Linux, Mac OS und andere Betriebssysteme geschrieben werden. Zur Implementierung der Clientseite werden HTML, CSS, JavaScript und Ajax verwendet. Zum Erstellen der Serverseite von Webanwendungen werden Programmiersprachen wie PHP, ASP, ASP.NET, Perl, C / C ++, Java, Python, Ruby und NodeJS verwendet. In dem vorherigen Schritt wurde es die Entscheidung getroffen, die Serverseite mit Web-Framework Django zu erstellen. Django wird mit einem leichten Webserver geliefert, mit dem eine Website schnell zum Laufen gebracht werden kann, ohne Zeit mit der Einrichtung eines Servers verschwenden zu müssen. Wenn der Entwicklungsserver von Django gestartet wird, überwacht er die Codeänderungen in Echtzeit. Es wird nach dem Ändern des Codes automatisch neu geladen.

In der zu realisierenden Abschlussarbeit ist auch notwendig einen weiteren Teil namens Register-Client zu entwickeln, an dem ein RFID-Leser angeschlossen wird. Der Register-Client verfügt selbst über keinen Datenbank und darf nur die abgelesene Daten dem Server schicken. Es geht um eine Simplex-Verbindung, da ein Nachrichtenverkehr asymmetrisch ist, weil der Register-Client keine Daten vom Server zurückbekommen darf und über den erfolgreiche oder gescheiterte Leihvorgang nicht wissen muss. Für die Implementierung des Register-Clients wird uComputer Raspberry Pi benutzt, der möglicherweise nicht der einzige Single-Board-Computer (SBC) auf dem Markt ist, aber bei weitem der beliebteste und schon zur Verfügung im PSE-Labor steht und ergänzend nicht geliefert werden muss. Der

Raspberry Pi wird von einer großen Anzahl von Menschen verwendet und viele offizielle und inoffizielle Ressourcen und Produkte umfasst - von Büchern und Zubehör bis zu Schulungen. Dank der Raspberry Pi Foundation werden regelmäßig neue SBC-Modelle veröffentlicht. Der Raspberry Pi eignet sich aufgrund seiner hohen Rechenleistung gut als Desktop-Computer und völlig für die Verwendung in oben geschrieben Zwecken passt .

2.1 Grundlagen der Webanwendungen

In diesem Kapitel werden die grundlegenden Begriffe zum Entwerfen und Erstellen eines Webanwendung erläutert. Hierbei liegt unser Fokus nicht auf den grafischen Aspekten der Browserfunktionalität (d. H. Dem Layout von Seiten, dem Rendern von Bildern). Stattdessen konzentrieren wir uns auf die Probleme im Zusammenhang mit der Verarbeitung von HTTP-Anfragen und -Antworten. Webanwendungen können abhängig von den verschiedenen Kombinationen ihrer Hauptkomponenten in verschiedene Bestandteilen unterteilt werden. Erstens wird Das Backend (Backend oder Serverteil der Anwendung) auf einem Remotecomputer ausgeführt, der von dem Endnutzer weit entfernt werden kann oder im anliegenden Raum stehen kann. Es ist in verschiedenen Programmiersprachen zu schreiben. Die Abschlussarbeit wird auf Python Sprache programmiert. Zweitens ist Das Frontend (Frontend oder Client-Teil der Anwendung) im Browser des Benutzers auszuführen. Die Webanwendung könnte theoretisch nur aus dem Client-Teil bestehen, wenn Benutzerdaten nicht länger als eine Sitzung gespeichert werden müssen. Dies aber ist nicht der Fall der Abschlussarbeit, da die Studentenkarten und der Verlauf des Verleihablaufs mindestens für ein laufenden Semester gespeichert werden muss, damit die Mitarbeiter des PSE-Labor immer eine Zugang zu allen gespeicherten vorherigen Leihvorgangs von der Ausleihe bis zur Rückgabe eines Boards. Es ist vorgesehen, dass am Ende des Semester nach dem letzte Rückgabe eines Boards die Datensätzen des zu Ende gegangen Semesters gelöscht wird.

Da in den letzten Jahren sich Webanwendungen rasant weiterentwickelt und die Desktop-Lösungen schrittweise ersetzt haben und sind zu einem wesentlichen Bestandteil des Geschäfts in der modernen Welt geworden haben, es sollte auch nicht unerwähnt bleiben, welche Vorteile die Webanwendungen haben:

- **Zugriff von jedem Gerät** Die Webanwendung kann überall auf der Welt von einem Computer, Tablet oder Smartphone zugegriffen und verwendet werden. Notwendig ist, dass dem Gerät eine Internetverbindung zur Verfügung steht.

- **die Kostenersparnis** Webanwendungen können auf allen Plattformen ausgeführt werden und müssen nicht mehr separat für Android und iOS entwickelt werden.
- **Anpassungsfähigkeit** Wenn native Anwendungen bestimmte Betriebssysteme erfordern, jedoch können jedes Betriebssystem (Windows, MAC, Linux usw.) und jeder Browser (Internet Explorer, Opera, FireFox, Google Chrome usw.) für die Arbeit mit einer Webanwendung.) verwendet werden.
- **Keine Software zum Herunterladen** Es ist günstig und einfach dem Endnutzer zu liefern, zu warten und zu aktualisieren. Das Aktualisieren auf die neueste Version erfolgt beim nächsten Laden der Webseite.
- **Netzwerksicherheit** Das Websystem verfügt über einen einzigen Einstiegspunkt, der zentral geschützt und konfiguriert werden kann.
- **Skalierbarkeit** Mit zunehmender Belastung des Systems ist es nicht erforderlich, die Leistung des Computer von Endbenutzer zu erhöhen. Mit einer Webanwendung kann in der Regel nur mithilfe von Hardwareressourcen eine größere Datenmenge verarbeitet werden, ohne den Quellcode neu zu schreiben und die Architektur ändern zu müssen.
- **Verhinderung von Datenverlust** Benutzerdaten werden in der "Cloud" gespeichert, für deren Integrität die Hosting-Anbieter verantwortlich sind, deswegen vom Verlust geschützt, falls die Festplatte des Computers beschädigt wird.

Als nächstes geht die Frage nach, wie Client und Server miteinander kommunizieren können. Der Client spricht mit dem Server über das HTTP-Protokoll. Das HTTP-Protokoll setzt die Verwendung einer Client-Server-Datenübertragungsstruktur voraus. Die Clientanwendung generiert eine Anforderung und sendet sie an den Server. Anschließend verarbeitet die Serversoftware diese Anforderung, generiert eine Antwort und sendet sie an den Client zurück. Die Clientanwendung kann dann weiterhin andere Anforderungen senden, die auf ähnliche Weise behandelt werden. Wenn ein Webserver eine Anforderung zum Bereitstellen einer statischen Webseite erhält, sendet er die Seite direkt an den Browser.[Ado] Wenn jedoch eine dynamische Seite angefordert wird, sind die Handlungsweise des Webservers nicht so einfach. Der Server übergibt die Seite an ein spezielles Programm, das die letzte Seite bildet. Ein solches Programm wird als Anwendungsserver bezeichnet. Der Anwendungsserver liest den Code auf der Seite, rendert die letzte Seite gemäß dem gelesenen Code und entfernt sie dann von der Seite. Als Ergebnis all dieser Operationen wird eine statische Seite erhalten, die an den Webserver übertragen wird, der sie wiederum an den Client-Browser sendet. Alle Seiten, die der Browser empfängt, enthalten nur

HTML-Code.[Ado] Schematische Darstellung des Prozesses kann man auf der Abbildung 2[Ado] ansehen. Die Anforderung kann mit den GET-Methoden erfolgen, wenn wir Daten empfangen möchten, und mit POST, wenn wir die Daten ändern möchten. Die Anfrage enthält auch den Host (Site-Domain), den Anfragetext (wenn es sich um eine POST-Anfrage handelt) und viele zusätzliche technische Informationen.



Abb. 2: Schematische Darstellung der Client-Server Kommunikation

Das HTTP-Protokoll ist ein zustandsloses Protokoll auf Anwendungsebene und bietet keine Speicherung von Informationen über die Sitzung des Benutzers; jede Datenübertragung wird als neue Sitzung betrachtet. Da HTTP per Definition ein zustandsloses Protokoll ist, wurde es nicht für die Unterstützung dauerhafter Verbindungen entwickelt. Eine Verbindung sollte lange genug dauern, damit ein Browser eine Anfrage senden und eine Antwort erhalten kann. Eine Verlängerung der Lebensdauer einer Anfrage darüber hinaus wurde nicht unterstützt.[RR03, p.62]

Am Rande sei auch eine weitere Technologie erwähnt, die uns ständig begegnet. Cache (Cache) ist ein Konzept in der Entwicklung, bei dem häufig verwendete Daten, anstatt jedes Mal aus der Datenbank abgerufen, berechnet oder auf andere Weise vorbereitet, an einem schnell zugänglichen Ort gespeichert werden. Einige Beispiele für die Verwendung des Caches:

- Django erhielt eine Anfrage, Daten für ein Diagramm in einem Bericht abzurufen. Daten aus der Datenbank werden abgeholt, vorbereitet und abgelegt in einer Datenbank mit schnellem Zugriff z.B. "memcached", die beispielsweise für eine Stunde zwischengespeichert wird. Bei der nächsten Anfrage werden die notwendigen Daten sofort von "memcached" erhalten und an das Frontend gesendet. Wenn es festgestellt wird, dass die Daten nicht mehr relevant sind, werden sie ungültig beziehungsweise gelöscht aus dem Cache.
- CDN-Anbieter (Content Delivery Network) werden zum Zwischenspeichern statischer Dateien verwendet. Hierbei handelt es sich um Servers auf der ganzen Welt, die für die Bereitstellung statischer Inhalte optimiert sind. Manch-

mal ist es effizienter, Bilder, Videos und JS-Skripte auf einem CDN anstatt auf Ihrem eigenen Server abzulegen.

- In allen Browsern ist das statische Zwischenspeichern von Dateien standardmäßig aktiviert. Da die Website nicht zum ersten Mal geöffnet wird, werden die dazugehörige Daten aus dem Zwischenspeichern viel schneller geladen. Der Nachteil für den Entwickler ist, dass bei aktiviertem Cache die vorgenommenen Änderungen nicht immer sofort sichtbar sind.

Daraus lässt sich die Schlussfolgerung ziehen, dass eine Webanwendung für einen Endnutzer wie eine Website aussieht, auf der die Webseiten mit teilweise oder vollständig nicht formatiertem Inhalt sich befinden. Die Endfertigung des Inhalts findet nur dann statt, nachdem ein Website-Besucher die Seite vom Webserver angefordert hat.

In diesem Kapitel wurden die Grundlagen der Webanwendungen und des HTTP-Protokolls erörtert. Diese Diskussion war nicht als umfassende Beschreibung aller Funktionen gedacht, eher als Überblick über das Verständnis und die Arbeit mit aktuellen Stand der Technologie und zukünftigen Verwendung der entwickelten Software im PSE-Labor.

2.2 Raspberry Pi Board und Betriebssystem

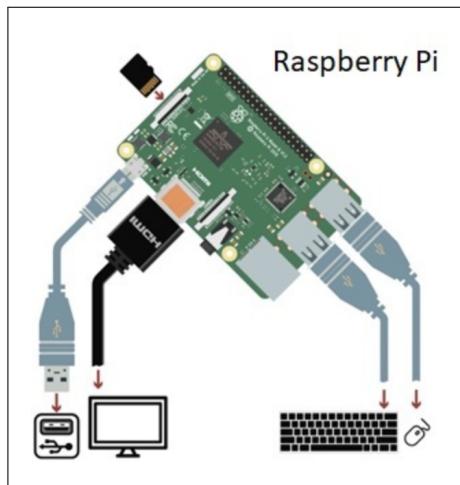


Abb. 3: Peripherieanschlüsse des Mikrocomputers

Raspberry Pi ist ein Einplatinencomputer, wobei die verschiedene Teile des Computers, die sich normalerweise auf separaten Platten befinden, werden hier nur auf einer dargestellt. Wie die meisten Einplatinencomputer ist der Raspberry Pi so klein wie eine Kreditkarte. Jedoch bedeutet das nicht, dass er nicht leistungsstark ist: Ein Raspberry Pi kann alles, was ein größerer und leistungsfähiger Computer kann, aber nicht unbedingt gleich schnell. Der Produktnamen kombiniert Raspberry-Himbeer- und Pi- die Kaiszahl Pi. Das Himbeerbild wurde zum Logo des Projekts. Der Raspberry Pi ist eine kostengünstige Plattform - sein empfohlener

Verkaufspreis beträgt weniger als 50€. Ein Mikrocomputer verfügt über alle Funktionen eines Personal Computer (PC): Prozessor, Speicher, Betriebssystem, Anschluss

an einen Monitor (TV), die Vernetzung. Der Raspberry Pi verfügt im Gegensatz zu einem PC über zusätzliche Peripheriegeräte wie GPIO-Ports (General Purpose Input / Output). Über diese Pins kann der Mikrocomputer mit der elektronischen Welt der Sensoren, Bildschirmen und Aktoren interagieren. In Abbildung 3[Pri] sind die Anschlüsse schematisch dargestellt.



Abb. 4: The Raspberry Pi's system-on-chip (SoC)

Für die Entwicklung der Abschlussarbeit und die Verwendung im Labor wird Raspberry Pi 3 Model B+ benutzt, an dem der RFID Leser angeschlossen wird. Das Modell "Raspberry Pi 3 B" ist eine kontinuierliche Weiterentwicklung zum Vorgängermodell "Raspberry Pi 2 B". Der Raspberry Pi 3 enthält einen Quad-Core-Prozessor mit 1,2 GHz von Broadcom und einen SDRAM-Arbeitsspeicher mit 1 GByte. Wie jeder Computer besteht der Raspberry Pi aus verschiedenen Komponenten, von denen jede eine Rolle in der Zusammenarbeit spielt. Die erste und wohl wichtigste davon befindet sich direkt über dem Mittelpunkt auf der Oberseite der Platine (Abbildung 1-2) und ist mit einer Metallkappe bedeckt: dem System-on-Chip (SoC). Der Name System-on-Chip ist ein guter Indikator dafür, was wird gefunden, wenn die Metallabdeckung abgenommen wird: einen Siliziumchip, der eine integrierte Schaltung ist und den Großteil des Raspberry Pi-Systems enthält. Dazu gehören die Zentraleinheit (CPU), die gemeinhin als „Gehirn“ eines Computers bezeichnet wird, und die Grafikverarbeitungseinheit (GPU), die die visuelle Seite der Dinge übernimmt. [Hal19, p. 11]

Auf der Unterseite des Raspberry Pi befindet sich einen weiteren Chip, der wie ein kleines schwarzes Plastikquadrat aussieht. Dies ist der Direktzugriffsspeicher (RAM) des Mikrocomputers. Wenn am Pi gearbeitet wird, wird die laufende Arbeit auf dem RAM gespeichert. Nur wenn die Daten explizit auf die microSD-Karte geschrieben werden, dann werden sie zum nächsten Verwendung gespeichert und nach dem Ausschalten nicht gelöscht. Zusammen bilden diese Komponenten die flüchtigen und nichtflüchtigen Speicher des Pi: Der flüchtige RAM verliert seinen Inhalt, wenn der Pi ausgeschaltet wird, während die nichtflüchtige microSD-Karte seinen Inhalt behält. Der Raspberry Pi erlaubt den Austausch des aktuellen RAM-Chips nicht. Der aktuelle RAM-Chip ist als BGA-Gehäuse direkt auf die Oberseite der CPU gelötet, was das Entfernen und / oder Ersetzen sehr schwierig macht, selbst wenn Sie den kompatiblen größeren Speicherchip finden könnten. Die Aufbau des Pi Der unterstützt den Austausch des aktuellen RAM-Chips nicht. Der aktuelle RAM-Chip ist als BGA-Gehäuse direkt auf die Oberseite der CPU gelötet. Das Hartlöten macht das Entfernen und / oder Ersetzen des RAMs sehr schwierig und deshalb es ist beim Kauf des bestimmtes Board auf die Größe des RAMs zu achten.

Die Besonderheiten des Raspberry Pi 3 ist, dass WLAN nach IEEE 802.11 b/g/n und Bluetooth Low Energy onboard sind und nicht durch externe USB-Adapter nachgerüstet werden müssen. Es ist für die vorgesehene Aufgabe wichtig ist, da die erforderlichen Komponenten nicht zusätzlich gekauft werden muss, was das Budget der PSE-Labors erspart und eine gewisse Zeit nicht geplant muss, dass die zusätzlich gekauften Komponenten irgendwie zum funktionieren bringen. Hier gab es in der Vergangenheit immer wieder Schwierigkeiten mit der Hardware-Erkennung oder Treiber-Probleme.

Die ARMv8-Architektur enthält Cortex-A53-Rechenkerne, die bei gleichem Takt schneller sind als die alten Cortex-A7-Kerne des Raspberry Pi 2 B. Von der 64-Bit-Fähigkeiten der ARMv8-Architektur profitiert allerdings nur neue Software, da 64 Bit auf der Hardware-Seite allerdings vom Betriebssystem und der Software auch unterstützt werden muss. Die folgende Liste gibt es eine kurzen Ansicht auf die Architektur des Mikrocomputers.

- System-on-Chip: BCM2837 64 Bit ARMv8 von Broadcom
- Prozessor: Quad-Core-Prozessor mit 1,2 GHz
- GPU: Dual-Core-GPU VideoCore IV mit OpenGL ES 2.0 und OpenVG mit Hardwarebeschleunigung und 1080p30 H.264 High-Profile-Decoding
- Arbeitsspeicher: 1 GByte LPDDR2-SDRAM
- WLAN: BCM43143 onboard für IEEE 802.11b, g und n im 2,4 GHz-Bereich
- Bluetooth: Bluetooth Classic und Low Energy (BLE) onboard (Bluetooth 4.1)

Eines der beliebtesten Betriebssysteme für den Raspberry Pi ist das Raspbian-Betriebssystem. Das Raspbian-Betriebssystem Raspbian basiert auf der ARM-Version von Debian 8 Jessie, ist für die Raspberry Pi-Hardware optimiert und enthält die Standardprogramme wie die LibreOffice Office Suite, einen Webbrowser, Claws Mail, eine leichtgewichtige Desktop-Umgebung und einige Programmier-Lernwerkzeuge. Neuere Versionen des Betriebssystems Raspbian verfügen über einen völlig modernen Chromium-Browser, mit dem auch komplexe Webseiten korrekt angezeigt werden können. Um eine Information auf einem großen Bildschirm anzuzeigen, wird Chromium nur im Vollbildmodus gestartet, den Mauszeiger ausgeblendet und den Bildschirmschoner ausgeschaltet. Es gibt verschiedene Möglichkeiten, Raspbian auf einem Raspberry Pi 3 zu installieren. Die erste besteht darin, das Dienstprogramm NOOBS zu verwenden, die zweite darin, den Inhalt des Bildes des Betriebssystem direkt auf die Karte zu schreiben. Während der Entwicklung des Register-Clients wurde

das Betriebssystem auf MicroSD-Karte geschrieben. Das Raspbian-Betriebssystem bootet von einer Micro-SD-Karte und das gesamte Betriebssystem läuft von der Karte.

Nach der kurzen Beschreibung der Raspberry Pi Architektur und sein Betriebssystem Raspbian ist schlusszufoltern, dass die Verwendung eines Raspberry Pi Mikrocomputers für die Implementierung des Register-Klient mit dem angeschlossenen RFID Leser ist eine lohnenswerte Entscheidung für in dieser Abschlussarbeit geschriebenen Aufgabe.

2.3 Kontaktlose Smartcards MIFARE

Anfänglich war die Smartcard eine Plastikkarte im ID-1-Format mit einer Größe von $85,60 \times 53,98$ mm und abgerundeten Ecken (eine Standardbank / Kreditkarte hat die gleiche Form und Abmessungen). Darin ist ein Mikrochip eingebettet, dessen Kommunikationskontakte zu einer Seite herausgeführt werden. Später erschienen Smartcards im ID-000-Format, sie wurden in Mobiltelefonen verwendet und sind uns als SIM-Karten bekannt. Dann erschienen Smartcards ohne externes Kontaktfeld, und sie begannen, einen Funkkanal für die Kommunikation und Energieübertragung zu verwenden. Smartcards haben keine eigene Stromquelle und sind auf eine externe angewiesen. Der Hauptvorteil einer Smartcard ist die physische Sicherheit der darauf gespeicherten Daten. Der Mikrochip ist sehr klein und alles notwendiges befindet sich darauf, ohne dass interne Kontakte hergestellt werden müssen, an die es sich zum Auffangen angeschlossen werden kann.

Der ungefähre Lebenszyklus einer Smartcard (z. B. einer Bankkarte mit Chip) besteht aus mehreren Phasen. Daran sind der Chiphersteller, der Smartcard-Hersteller, der Kunde und der Kunde beteiligt:

- **Herstellung von Chips/Prozessoren.** Zu diesem Zeitpunkt schreibt der direkte Hersteller des Chips nach der physischen Produktion die vom Hersteller der Smartcards bereitgestellte universelle und für alle Karten des gleichen Typs dieselbe Firmware auf.
- **Initialisierung der Smartcard.** Die Chips werden an den Kartenhersteller gesendet. Er ändert die Firmware nach Bedarf. Beispielsweise schreibt er eine eindeutige Seriennummer in jede Karte. Anschließend deaktiviert er die Möglichkeit, diese mit einer speziellen Anforderung zu ändern.

MIFARE® contactless tag IC family overview												
Product features	MIFARE Ultralight®			MIFARE Classic®		MIFARE Plus®			MIFARE® DESFire®			
	Nano	EV1	C	EV1	SE	EV2	Light	EV3	EV2			
RF Interface	ISO/IEC 14443-3				ISO/IEC 14443-3, Type A 13.56 MHz				ISO/IEC 14443-4			
Protocol	7-byte UID				7-byte UID, 4-byte NUID, Random ID				7-byte UID, Random ID			
UID - unique identifier	106 Kbps				1K				106-848 Kbps			
Communication speed	40	48	128	144	4K	1K	2K	4K	640	2K	4K	8K
Memory size [Bytes]									Pre-configured file system			16K
Memory model	Compact, 4-byte pages				Compact, sectors & 16-byte blocks				Flexible file system			
Crypto	-		3KDES	Crypto-1	48-bit	Crypto-1, AES	48-bit Crypto-1, 128-bit AES	128-bit AES	AES/LRP	DES/3K3DES/3K3DES/AES		
Key length	-		112-bit							128-bit AES, up to 168-bit DES		
Authentication	-	Password							3-pass mutual			
Communication security				Encrypted					Plain, CMACed, encrypted w. CMAC			
MissmartApp												✓
Transaction MAC	-									✓		
Transaction Timer	-					✓						
Security Level upgrade	-			card		sector per sector						
SL1513MxMode					✓							
Multi key sets												✓
Proximity check						✓						✓
Virtual card concept						✓						✓
Restrict update operations in SL1						✓						
Originality check features	ECC signature programmable	ECC signature	-	ECC signature	-	AES originality keys	AES originality keys, ECC signature	✓		AES originality keys, ECC signature		
CC Certification							EAL5+	EAL4		EAL5+		
ISO 7816-4 APDU	-					✓						
NFC compliance	NFC Forum type 2 tag compliant			Not supported by majority of NFC devices		NFC capable in SL3	NFC capable in SL1 and SL3	NFC Forum type 1 tag V2.0 compliant				
Target application	Public transport & event ticketing, loyalty programs, limited use tickets			Single application - not recommended for new design		Public transport/ campainards/ access management		Smart city platform/ advanced mobility multi-applications/ micropayment/ loyalty programs/ access management				
Input capacitance [pF]	17/50			17	17/70		Supported via MAD	Supported via MAD	17/50	17/70		
Multi applications							Fixed, single application					Dynamic

Abb. 5: Mifareproduktfamilie

- **Smartcard-Herstellung.** Der Hersteller legt den Chip in Karten des gewünschten Formats ein und sendet diese an den Kunden, beispielsweise eine Bank.
- **Personalisierung.** Der Kunde schreibt unter Verwendung der Methoden der Firmware auf der Karte seine Anwendungen darauf, z. B. Bankgeschäfte, sowie zusätzliche Daten, z. B. Kundenname, Kontonummer usw. Danach wird die Karte durch eine spezielle Anfrage finalisiert, wonach beispielsweise die Aufzeichnung neuer Anträge eingeschränkt wird.
- **Ausgabe.** Die Karte wird dem Kunden ausgegeben und vom Kunden verwendet.
- **Zerstörung.** Die Karte wird weggeworfen.

Einer der größten Hersteller von Smartcards ist die Firma NXP Semiconductors mit Hauptsitz in den Niederlanden. Mit einer großen Produktricke schafft diese Lösungen für kontaktlose Zugangs- und Zeitkontrollen und sichere kontaktlose Automobilzugangskontrollen. Die weltweit meistgenutzte kontaktlose Chipkartentechnik „MIFARE“ wurde von NXP Semiconductors entwickelt und die Produktfamilie umfasst mittlerweile vier Produktreihen. (siehe Abbildung 5) [NXP] Alle MIFARE-ICs sind konform zur Norm ISO/IEC 14443 und erfüllen somit die Standards für die kontaktlose Kommunikation zwischen Chipkarten. Sie arbeiten mit 13,56 MHz und ihre Leseentfernung ist bis 10 cm möglich. Kontaktlose MIFARE-Smartcards verfügen über einen 1-KByte-Speicher, der in 16 Sektoren mit jeweils 16 Byte unterteilt ist. Die Datenspeicherdauer beträgt bis zu 10 Jahre. Die Anzahl der Umschreibzyklen beträgt 100.000 Zyklen.

Neben NXP Semiconductors werden Chips für Mifare-Smartcards von der deutschen Firma Infineon im Rahmen einer Lizenzvereinbarung hergestellt[Too]. Nur Smartcards mit Chips von NXP Semiconductors und Infineon dürfen die Marke Mifare in ihrem Namen tragen. Nur Karten mit diesen Chips können als "Original" bezeichnet werden.

Die Studentenkarte der Beuth Hochschule wurden mit Mifare DESfire EV1 Kartenchip hergestellt. MIFARE DESFire EV1 ist die nächste Generation von Mifare Desfire mit einigen verbesserten Funktionen der Sicherheit und Verschlüsselung[Chi14, p.83]. Unberechtigte können sie aufgrund der AES-Verschlüsselung nicht auslesen. Zudem enthalten die Karten elektronisch keine persönlichen Daten[TBb]. Alle auf der Karte gespeicherte Daten werden kodiert, z.B. mit der UID der einzelnen Karte verschlüsselt. MIFARE DESFire wird in vielen NFC- und RFID-Anwendungen verwendet, da er als sicherer Transponder mit einem von drei verschiedenen Typen von Verschlüsselung gesichert ist: Single DES, Triple DES oder AES. Im Allgemeinen wird AES als die sicherste Verschlüsselungsstufe der oben aufgeführten Methoden angesehen. Der AES-Authentifizierungsprozess besteht aus mehreren Schritten, in denen der NFC / RFID-Reader und das MFDFEV1-Tag verschlüsselte Daten austauschen, um zu überprüfen, ob sie denselben Schlüssel verwenden. Während dieses Vorgangs wird ein Sitzungsschlüssel erstellt, der für bestimmte Befehle wie den ChangeKey-Befehl verwendet wird[JI].

Zusammenfassend lässt sich sagen, dass die neue Studentenkarte, die an der Beuth Hochschule ab Sommersemester 2018 verwendet wurden, eine zuverlässige und zeitgemäße Lösung. Da die Karte elektronisch keine persönlichen Daten der Studierenden beinhaltet und die Automaten alle persönlichen Daten anhand eines Pseudonyms online abrufen müssen (d.h. liegen in den Automaten auch keine persönlichen Daten vor), es in Fall des Verlusts die persönliche Daten von den Unberechtigten nicht ausgelesen werden können[TBb]. Das stand im Fokus der Entscheidung, eine Studentenkarte als einzige elektronischer Identifizierungsmittel beim Ausleihe/Rückgabevorgänge im PSE-Laboz zu benutzen.

2.4 Sender-Empfänger-System mit RFID

Im vorherigen Kapitel 2.3 wurden die Grundlagen und Vorteilen der MIFARE-Technologie für Smartcards erklärt. Im aktuellen Kapitel wird über die Sender-Empfänger-System mit RFID beschrieben. Für die Implementierung der Aufgabe wird sowohl MIFARE-Smartcards als auch RFID-Tag-Mikrochips verwendet. Der letzte wird zum Identifizierung der auszuleihenden Raspi-Boards benutzt und auf der Rückseite des jeden Boards unter dem Schutzschirm geklebt werden. Beide Typs

können mit dem RFID-Leser ACR122U-A9 von Advanced Card Systems abgelesen werden. Ab hier wird weiter nur den Begriff "Tag" sowohl für MIFARE-Smartcards als auch für RFID-Tag-Mikrochips verwendet und den Unterschied in Namen nicht weiter angemerkt. Die gespeicherte Daten warten darauf, gelesen zu werden. Die Antenne des Tags erhält Energie von einer RFID-Leseantenne. Mit der Stromversorgung der internen Batterie oder des Lesegeräts sendet das Tag Funkwellen an das Lesegerät zurück. Der Leser nimmt die Funkwellen des Tags auf und interpretiert die Frequenzen als Daten. RFID-Tags, die über einen Teil des elektromagnetischen Spektrums gesendet werden, und die genaue Frequenz können ausgewählt werden, um Interferenzen mit anderen elektronischen Geräten zu vermeiden. Der Leser sendet ein Signal an das Tag und liest seine Antwort[Tec]. Der Leser verfügt über einen Funkempfänger, der als Transceiver bezeichnet wird und ein codiertes Funksignal an das Tag sendet. Das Signal aktiviert das Tag und der Transponder wandelt das Signal dann in eine nutzbaren Leistung um und sendet auf den Leser. Das Tag empfängt die Nachricht und antwortet dann mit seiner Identifikation und anderen Daten. Dies kann eine eindeutige Seriennummer des Produkts oder produktbezogene Daten sein. In der Abschlussarbeit wird nach der UID des Tags gefragt.

Während sich jedes Sender-Empfänger-System mit RFID hinsichtlich Gerätetyp und Komplexität unterscheidet, enthält jedoch jedes RFID-System mindestens die folgenden vier Komponenten: Leser, Antennen, Tags und Kabel. Das einfachste System kann aus einem mobilen Hand-RFID-Lesegerät (mit integrierter Antenne) und RFID-Tags bestehen, während komplexere Systeme mit Multi-Port-Lesegeräten, GPIO-Boxen, zusätzlichen Funktionsgeräten, mehreren Antennen, Kabeln und RFID-Tags ausgelegt sind und ein komplettes Software-Setup benötigen können. RFID-Tag-Typ Bestimmt das RFID-System. Derzeit gibt es drei verschiedene Arten von Tags: **aktiv, semi-passiv und passiv**. Ein aktives Tag sendet



Abb. 6: Verschiedenen RFID-Tags

mit seiner Batterie Radiowellen an einen Leser, während eine semi-passive Tag-Batterie in Gegenwart eines Lesers aktiviert wird. Aktive und semi-passive Tags werden über größere Entferungen gelesen. Sie senden hohe Frequenzen von 850 bis 950 MHz, die von 30 Meter oder mehr gelesen werden können. Zusätzliche Batterien können die Reichweite eines Tags auf über 90 Meter erhöhen. Passive RFID-Tags haben keine Batterie und verwenden die vom Lesegerät übertragene Funkenergie als Stromquelle. Diese Tags werden bis zu ein paar Meter entfernt

gelesen und sind kostengünstiger[Tec]. Für die Markierung der Raspi-Boards im PSE-Labor werden die passive RFID-Tags verwendet, die für vorgesehenen Zwecken ideal dienen. Die runde benutzte im Abschlussprojekt RFID-Tags sind auf der Abbildung 6 im Vergleich in der Größe zu 1-Euro Münze zu sehen. RFID-Systeme können nach Tag- und Lesertyp klassifiziert werden: passiver Leser - aktiver Tag (PRAT), aktiver Leser - passiver Tag (ARPT) und aktiver Leser - aktiver Tag (ARAT). Das ARPT-System wird im PSE-Labor eingesetzt und verfügt über einen aktiven Leser und empfängt Authentifizierungssignalantworten von passiven Tags. Ich möchte an dieser Stelle auch noch anmerken, dass der Initialisierungsprozess für eine kontaktlose Karte viel komplizierter als für eine Kontaktkarte ist. Das meiste davon wird vom sogenannten Antikollisionszyklus besetzt. Eine Kollision tritt auf, wenn mehr als eine Karte gleichzeitig auf das elektromagnetische Feld der RFID-Leser trifft und der diese Karten voneinander unterscheiden muss. Der Algorithmus dieses Prozesses ist sehr komplex und umfasst mehrere zehn Seiten Beschreibung in den Normen ISO/IEC 14443-2 und ISO / IEC 14443-3. Daher werde ich ihn hier nicht angeben, da es nicht von Entwickler wirklich etwas zu machen benötigt wird - das Terminal und der RFID-Leser sind voll damit beschäftigt.

An dieser Stelle muss es gesagt werden, dass die RFID-Tags bestimmte Nachteile haben, die wurden zwischen den Mitarbeiter und Autorin der Abschlussarbeit diskutiert. Es wurde jedoch die Entscheidung getroffen, dass RFID-Tags für die vorgesehenen Zwecken des Verfolgen des Raspi-Boards (welchen Raspi-Board von welchem Student am welchen Tag ausgeliehen wurde und bis zum welchen Tag zurückgegeben muss) die Sicherheitserwartungen der PSE-Labor Mitarbeiter erfüllen. RFID-Tags sind aus mehreren Gründen im Vergleich nicht ideal. Da ein RFID-Tag nicht zwischen Lesegeräten unterscheiden kann, können die Informationen von fast jedem gelesen werden, sobald sie die ursprüngliche Lieferkette verlassen haben. Weil RFID-Lesegeräte so tragbar sind und die Reichweite einiger Tags so groß ist, können Betrüger Informationen sammeln, auf die sie sonst keinen Zugriff hätten. Dies bedeutet, dass jeder ohne Wissen einer Person potenziell sensible Informationen sammeln kann. Diese Nachteile der RFID-Tags wurden vernachlässigt, da sowohl Studentenkarte als auch geklebte auf den Raspi-Boards RFID-Tags keine sensible Information behalten. Es gibt trotzdem ein Gefahr, dass entweder die Studentenkarte geklont von einem Täter wird, um sich für einen Student ausgeben und ein Raspi-Board stehlen zu können, oder ein Raspi-Board geklont von einem Täter wird, um ein Datensatz in der Datenbank zu erzeugen, dass schon ausgeliehenen Board quasi zurückgegeben wurde, obwohl in der Realität den Raspi-Board nie zurückgegeben wurde. Gegen das Klonen des Raspi-Tags wurde es besprochen, dass in der Zukunft im PSE-Labor im Schrank mit den Raspi-Boards jeden Platz für jeden entsprechenden Raspi-Board mit einem Gewichtssensor ausgerüstet werden wird und acaLoan-System auf der Erscheinung des bestimmten Gewicht erwarten wird. Dies ist aber nicht der Teil bestehenden Abschlussarbeit und von Mitarbeiter des

PSE-Labor als eine spannende Aufgabe für die andere Abschlussarbeit vorgesehen ist. Gegen das Klonen des Studentenkarte wurde es zuerst entschieden, dass ein bestehenden Zugang zu einem Schrank mit Raspi-Boards entlang die beide Arbeitstischen der Mitarbeiter des PSE-Labor eine bestimmte Sicherheit gewährleisten könnte. Nach anderen Lösungen wird es weiter noch diskutiert und es liegt außer den Rahmen der bestehenden Abschlussarbeit.

2.5 HTTP für Design der verteilten Systeme

2.6 Django Framework

Django ist ein Open Source-Webentwicklungsframework, das in der Python-Sprache geschrieben ist. Es wurde entwickelt, um so viele Prozesse wie möglich zu automatisieren, sodass es sich auf die Softwareentwicklung konzentriert werden können, ohne Zeit mit dem Einrichten des Servers verschwenden zu müssen. Django wurde ursprünglich so konzipiert, dass es lose Kopplung zwischen verschiedenen Teilen der Infrastruktur hat, sodass es unabhängig voneinander gearbeitet werden kann. Diese Unabhängigkeit bedeutet, dass es in der Entwicklungsprozess nur die Teile von Django verwendet werden kann, die benötigt werden, ohne sich um Probleme mit der Interdependenz von Komponenten kümmern zu müssen. Durch die Verwendung von Django wird die erforderliche Codemenge reduziert, wodurch das Schreiben von Webanwendungen schneller und die Wartung Ihrer Anwendung in Zukunft erheblich vereinfacht wird. Django folgt strikt dem DRY-Prinzip (Don't Repeat Yourself), dass jeder einzelne Code oder jede einzelne Daten an nur einem Ort gespeichert werden sollte. Dies vereinfacht und beschleunigt den Softwareänderungsprozess erheblich, da eine Anwendung, die geändert werden muss, an einem einzigen Ort ausgeführt werden muss.

Nach der Django Instalation und dem Erstellen eines Projekts ist es nützlich, die erstellte Projektstruktur zu betrachten[Foua]:

- **manage.py** ist ein Befehlszeilenprogramm, mit dem auf verschiedene Weise mit Django-Projekt interagiert wird . Diese Datei muss nicht geändert werden.
- **acaLoanRaspiBoard** Projektarbeitsverzeichnis enthält:

`__init__.py`: Eine leere Datei, die Python mitteilt, dass dieses Verzeichnis für das Python-Paket bestimmt ist.

settings.py: Die Einstellungen und Konfigurationsdatei für das Django-Projekt.

urls.py: Die URL-Beschreibungsdatei für dieses Django-Projekt.

asgi.py: Damit kann die Anwendung mit einem Webserver unter Verwendung des ASGI-Protokolls arbeiten.

Die generierte Datei *settings.py* enthält eine Grundkonfiguration für die Verwendung einer SQLite-Datenbank und eine Liste von Django-Anwendungen, die standardmäßig zum Projekt hinzugefügt werden sollen. In der Datei *settings.py* ist es möglich, den Debug-Modus des Projekts zu aktivieren / deaktivieren. Wenn "Debug=true" festgelegt ist, zeigt der Server bei Auftreten einer nicht erfassten Ausnahme detaillierte Fehlermeldung an. Es muss beim Wechsel zur Produktionsversion auf "false" gesetzt, da mit aktiviertem Debugging alle Benutzer vertrauliche Projektdaten sehen können.

Die Architektur von Django basiert auf den Ideen des The Model-View-Controller (MVC) Design Pattern, womit Anwendungsschicht, Benutzeroberfläche (UI) und Datenzugriffsebenen getrennt werden, sodass jede dieser Ebenen unabhängig von anderen geändert werden kann. Als Konzept ist das MVC-Entwurfsmuster wirklich einfach zu verstehen[Geo17, pp. 15-16]:

- **Das Modell (M)** ist ein Modell oder eine Darstellung Ihrer Daten. Es handelt sich nicht um die tatsächlichen Daten, sondern um eine Schnittstelle zu den Daten. Mit dem Modell können Sie Daten aus Ihrer Datenbank abrufen, ohne die Feinheiten der zugrunde liegenden Datenbank zu kennen. Das Modell stellt normalerweise auch eine Abstraktionsschicht mit Ihrer Datenbank bereit, sodass Sie dasselbe Modell mit mehreren Datenbanken verwenden können.
- **Die Ansicht (V)** ist das, was mit den Augen gesehen werden kann. Dies ist die Präsentationsebene für das Modell. Auf dem Computer wird die Ansicht im Browser für eine Web-App oder in der Benutzeroberfläche für eine Desktop-App angezeigt. Die Ansicht bietet auch eine Schnittstelle zum Sammeln von Benutzereingaben.
- **Der Controller (C)** steuert den Informationsfluss zwischen dem Modell und der Ansicht. Mithilfe der programmierten Logik wird entschieden, welche Informationen über das Modell aus der Datenbank abgerufen und welche Informationen an die Ansicht übergeben werden. Es erhält auch Informationen vom Benutzer über die Ansicht und implementiert die Aufgaben der Anwen-

dung: entweder durch Ändern der Ansicht oder durch Ändern von Daten über das Modell oder durch Ändern die beiden Elementen.

2.7 Grundlagen der REST API

2.8 Clientseitiges JavaScript

Systemdesign

Die Softwareentwicklung beginnt mit einer Beschreibung der Bedürfnisse und ihrer Analyse. Je genauer und korrekter die Beschreibung der Softwareanforderungen und deren Analyse ist, desto einfacher ist es, alle nachfolgenden Schritte abzuschließen. Das Hauptproblem in dieser Phase ist der Unterschied in den Ansichten des Kunden (in dem Fall der vorliegenden Abschlussarbeit sind die Kunden die PSE-Labor Mitarbeiter) und des Entwicklers (die Autorin der Abschlussarbeit). Es wurde auch die Entscheidung getroffen, mit welchen Hardware ist die Aufgabestellung zu implementieren. Jedoch während der Entwicklung der Register-Client (der einer von drei Bestandteilen der Software, an dem ein RFID-Leser angeschlossen werden muss), wurde schließlich ein RFID-Leser gewechselt. Der Fall ist im Kapitel 4.1.2 nachzulesen. Im Rahmen der Analyse und des Systemdesigns wurden die Struktur und Zusammenhänge der Elemente des zu entwickelnden Systems untersucht. Das Ergebnis dieser Untersuchung enthält genügend Informationen, um das System zu implementieren und ist unten in folgenden Kapitels detailliert beschrieben.

3.1 User Stories

Zuerst wurden die User Stories erstellt, die eine diskutierte Darstellung der Absicht (Endbenutzer muss/will so etwas tun) zeigen können. Es ist mithilfe der User Stories zu klären, was die zu entwickelnde Datenbank leisten soll. Es wird auf die Frage konzentriert, welche Daten in der Datenbank gespeichert werden sollen. Der Text der User Story selbst sollte die Rolle / Aktionen des Benutzers im System, seine Bedürfnisse und den Gewinn erläutern, den der Benutzer nach dem Auftreten der Story erhält. Zum Beispiel: Wie *<Benutzerrolle / Charakter>, ich <möchte etwas bekommen>, <für diesen und jenen Zweck>*. Während des Schreibens der User Story wurden zwei Gruppe von Stakeholders definiert: die Studierende (Beuth Studentinnen und Studenten) und der Admin (PSE-Labor Mitarbeitern). Es wurden die folgenden User Stories erstellt, die später während der Entwicklungsphase implementiert wurden:

- As a student, I want **to loan a board** so I can work at lab
- As a student, I want **to loan a board** so I can work at home
- As a student, I want **to list boards assigned to me** so that I am sure I to return

- As a student, I want **to return a board from lab work** so that I can loan again
- As a student, I want **to return a board from home work** so that I can loan again
- As a student, I want **to initiate session** so I can loan a board
- As a lab admin, I want **to mark a board** so it can be loaned for home or for lab
- As a lab admin, I want **to terminal session** with timeout so that students can use the loan station
- As a lab admin, I want **to block a student** so that they won't be able to loan a board
- As a lab admin, I want **to see all students** registered on the course so that I can manage their profiles
- As a lab admin, I want **to see all loaned boards** so that I can know their expected return date
- As a lab admin, I want **to register students** so that they are able to loan boards
- As a lab admin, I want **to register new boards** so that they can be loaned
- As a lab admin, I want **to delete student's record** when semester ends so that they are not stored anymore in database

3.2 Anforderungen

Während die meisten neuen Funktionen mithilfe der User Stories aus Anwendersicht definiert werden sollten, ist dies nicht immer machbar oder sogar hilfreich, wenn es zu Sicherheitsfunktionen oder Infrastrukturanforderungen kommt, die nicht kundenorientiert sind. Es gibt zwei Arten von Anforderungen: funktionale Anforderung und nichtfunktionale Anforderungen. Während der Analysephase wurden die beiden Arten von Anforderungen definiert.

3.2.1 Funktionale Anforderungen

Eine funktionale Anforderung beschreibt, was ein Softwaresystem tun sollte. Es werden die folgenden funktionalen Anforderungen definiert:

- The background color for all windows in the application will be white and have a hexadecimal RGB color value of 0x0000FF.
- The colors of design guidelines of Beuth Hochschule will be used.

- The software automatically validates whether a student is able to loan a board for a homework.
- The software automatically shows the information about boards that student already loaned.
- Student will see their name after they scanned their valid student card.
- Student will see board's number after they scanned a Raspberry Board.
- If student can not loan a board they will see an information message on a display.
- If a student can not loan a board the session should be terminated
- If an error during the loan process is occurred the student will see detailed information so they can later talk with an admin about the issue.
- Error states will be marked with red color on the page
- Succeeded states will be marked with a green color on the page

3.2.2 Nichtfunktionale Anforderungen

Nichtfunktionale Anforderungen bestimmen nicht die Funktionen, sondern die Eigenschaften des Systems: Leistung, Zuverlässigkeit, Verfügbarkeit, Skalierbarkeit und eine Reihe anderer Parameter, die das System einschränken und verbessern sollen.

- The Server has to be implemented using a modern Python web framework Django.
- The user Interfaces (frontend) shall be implemented as HTML pages with dynamic content inside based on Django Templates.
- The admin views must require authorization. A view decorated with this function will be executed normally only if the logged user has admin rights.
- The SQLite relational database will be used in order to store students, boards and loan records
- For terminating the session automatically after timeout the command line application (CLI) will be implemented using python and will be run on the server.
- Users must use for the initial login their student card. Moreover, every next login will be done with the same card.
- Students never allowed to loan home board longer than 1 week (7 calendar days). Such attempt should be reported to the security administrator.
- Students never allowed to loan lab board longer than 120 minutes. At the end of the exercise administrator should be notified if the board was not returned.

- Loan process can not be started if a student was not properly registered on the course
- Every unsuccessful attempt by a user to loan/return an item shall be recorded on an audit trail.
- Only one active session is allowed for loan/return process. No multi-user mode is intended
- If the current session is inactive longer than 180s the session will be terminated and should be restarted
- The actions made on RFID-Reader should be displayed on the screen with an acceptable delay for a human (less than 5 seconds)

Während die User Story selbst die Verbindung zwischen der menschlichen Wahrnehmung und der technischen Umsetzung ist, können mithilfe der Anforderungen die Betriebsfunktionen und Einschränkungen des Systems beschrieben werden, die seine Funktionalität verbessern.

3.3 Systemmodell mit UML

3.4 Komponentendiagramm

3.4.1 Klassendiagramm

Für das Systemarchitektur wird das UML-Klassendiagramm entwickelt, das einen Überblick über ein Softwaresystem bietet, indem Klassen, Attribute, Operationen und deren Beziehungen angezeigt werden. Dieses Diagramm enthält den Klassennamen, die Attribute und die Operation in separaten festgelegten Fächern. In der Entwurfsphase wird es festgelegt, welche Klassen das System benötigt wird. Die festgestellte Klassen werden weiter nicht wie üblichen Python-Klassen implementieren, jedoch wie eine Django Modelle direkt zum Erzeugen der Datenbank verwendet.

3.4.2 Anwendungsfalldiagramm

Während der Analyse der Systemanforderungen haben wir festgestellt, dass die Software aus vier zentralen Anwendungsfällen bestehen sollte. Diese sind: Die Ausleihe des Lab-Loan Boards, die Rückgabe des Lab-Loan Boards, die des Ausleihe des Home-Loan Boards, die Rückgabe des Home-Loan Boards. Jeder von diesen Anwendungsfälle kann erfolgreich oder mit dem Fehler beendet werden.

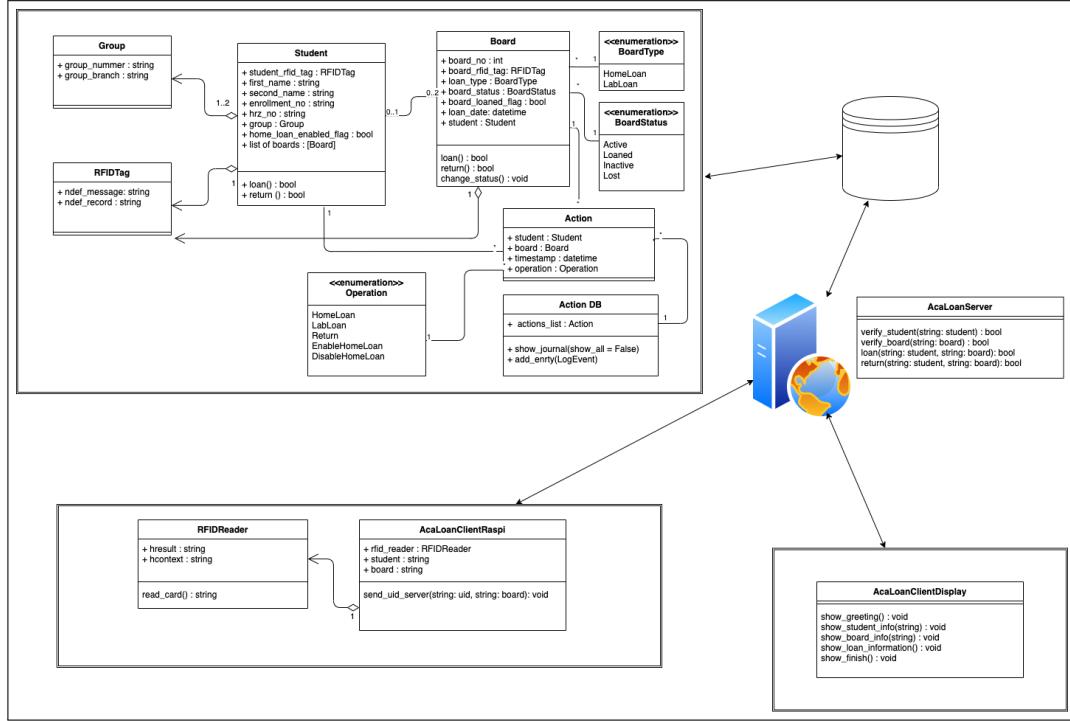


Abb. 7: UML Klassendiagramm.

3.5 Systemarchitektur

Die Architektur besteht aus den grundlegenden Beschreibungen (Ansichten), die zeigen, aus welchen grundlegenden Teilen (Komponenten, Modulen, Layouts usw.) das System besteht und wie diese Beschreibungen zusammenhängen. Das System aus Sicht des Benutzers umfasst Artefakte in Bezug auf praktische Anwendungsphasen, Szenarien und Workflows. Das System aus Sicht des Managers oder Kunden umfasst Artefakte, die die Grenze zwischen dem System und seiner Umgebung definieren, grundlegende Funktionen und Verhaltensweisen des Systems, sowie interne und externe Benutzer. Endlich das System aus Sicht der Designer definiert physische Sicht. Dies umfasst Artefakte, die die physische Grenze des Systems, Komponenten, ihre Interaktionen zwischen interne Datenbanken und Datenstrukturen und die bei seiner Entwicklung verwendeten Standards definieren.

Während des Entwurfs, der Entwicklung und der Modernisierung eines Softwaresystems erfordert die „Reihe von Entscheidungen über seine Organisation“ (Architektur) eine ständige Diskussion mit allen Beteiligten des Projekts. Auch hier ist es wichtig, dass jeder die gleiche Vorstellung über das System von sich hat.

Es ist offensichtlich, dass für die effektive Präsentation architektonischer Lösungen für alle interessierten Parteien eine bestimmte Methode erforderlich ist, die es ermöglicht, ihre Essenz einem möglichst breiten Personenkreis auf zugängliche,

aber ausreichend detaillierte Weise zu vermitteln. Ein Klassendiagramm kann auf diese Weise dienen. Das Klassendiagramm definiert die Objekttypen im System und die verschiedenen Arten von Beziehungen, die zwischen ihnen bestehen. Es bietet eine allgemeine Ansicht einer Anwendung. Diese Modellierungsmethode kann mit fast allen objektorientierten Methoden ausgeführt werden. Das Klassendiagramm erklärt, was sind die Komponenten des Systems und wo sollen wir die Einkapselungsbarrieren platzieren. Welche Entscheidungen sind innerhalb von Komponenten zu verbergen, damit sie geändert werden können, ohne den Rest des Systems zu beeinträchtigen. Wie und was müssen die Komponenten wirklich kommunizieren. Zum Beispiel, was sollte in den Schnittstellen sein oder welches Protokoll sollte verwendet werden? Auf der Abbildung 7 den Zusammenhang zwischen die drei Bestandteilen (Register-Client, Server und Display-Client) mit den Einkapselungsbarrieren zu sehen. Da es schon während der Besprechung der Aufgabestellung mit den PSE-Labor Mitbietern für eine Entwicklung mit Django Framework entschieden wurde, wird es im Klassendiagramm schon zu sehen. Die Klassen wurden am Anfang als die Django Modelle geplant, die in einer SQLite Datenbank verwaltet werden. Dann muss z.B. kein Primärschlüssel in Datenbankdesign definiert werden, da es von Django eine Primärschlüssel erzeugt wird und Entwickler darüber nicht kümmern muss. Die drei Komponenten werden miteinander über HTTP-Protokoll kommunizieren. Es wird mithilfe API-Endpunkt geschehen, an dem eine Verbindung mit den drei Bestandteilen der Softwareprogramm herstellt wird. An den API-Endpunkt werden Informationsanforderungen von einer Webanwendung einem Webserver geschickt die Antwort empfangen.

3.6 Endliche Zustandsmaschine

Implementierung

In meiner Abschlussarbeit präsentiere ich die praktische Lösung für das PSE-Labor der Beuth Hochschule für Technik Berlin. Die gesamte Aufgabe lässt sich in drei Bestandteile unterteilen: Register-Client, Server und Display-Client. Erstens wird Register-Client implementiert, damit wird RFID Leser am Raspberry Pi Mikrocomputer angeschlossen, alle Treiber installiert und auf Python Programmierung Sprache die Software geschrieben, die die ständige Überwachung des empfangenden von RFID Leser Daten zulässt und die Verbindung mit dem Server zulässt. Falls die empfangene Daten korrekt sind, d.h. eine richtige MIFARE Studentenkarte oder einen richtigen RFID-Transponder abgelesen wurde, schickt die Software die abgelesene Daten zum Server ab. Der Server ist der zweite Bestandteil der Abschlussarbeit und wird mit Hilfe Django Framework, Django Finite State Machine auf Python Programming Sprache implementiert. Server enthält die Datenbank mit die Datensätzen über die alle im PSE-Labor vorhandenen ausleihenden Boards, die zum Modul im laufenden Semester registrierten Studenten und geschehenen Ausleihe/Rückgabe-Vorgänge. Es wird von Server überprüft, ob eine von Register-Client abgelesene Studentenkarte einem zugelassenen für die Ausleihe Student gehört und die entsprechenden Information auf Display-Client geschickt. Es wird auch von Server bestätigt, ob für die Ausleihe/Rückgabe neben dem RFID-Leser gehaltenen Raspi Board dem Student ausgeliehen/vom Student zurückgegeben werden darf. Darauf aufbauend, wird der dritte Teil namens ein Display-Client als dynamische HTML-Seite realisiert, die eine Verbindung zum Server Mithilfe des HTTP-Protokolls und eingebauten im Browser Kommunikationsmittel die asynchrone Nachrichten zu schicken, bereitstellt. Für die dynamische Aktualisierung des Inhalts der Webseite und einen Zugang zum asynchronen HTTP-Client wird jQuery benutzt.

4.1 Register-Client

Das folgende Kapitel beschäftigt sich mit der Implementierung des Register-Client auf Raspberry Pi Board mit angeschlossenen RFID-Leser. Dieser Teil der verteilte System lässt sich wie folgendes unterteilen. Zuerst wurde das Betriebssystem Raspbian auf Board zum Leben gebracht und dann die alle notwendigen für RFID-Leser Treiber installiert. Nach dem der RFID-Leser funktionieren angefangen und die Daten von

RFID-Transponder abgelesen hat, wurde die nächste Herausforderungen gelöst: die Struktur die zu empfangenen Daten wurde verstanden, richtig bearbeitet, eine JSON-Datei erstellt und durch die HTTP-Protokoll dem Server geliefert.

4.1.1 Installation des Betriebssystem

Der vorhandene für die Abschlussarbeit Raspberry Pi 3 Model B+ wurde nicht als Starter Kit mir übergeben, dann wurde es zusätzlich benötigt[Hal19, pp. 21-22]:

- **USB-Netzteil** mit einer Nennleistung von 2,5 A (2,5 A) oder 12,5 Watt (12,5 W) und einem Micro-USB-Anschluss.
- **microSD-Karte**, die als permanenter Speicher des Raspberry Pi dient; Alle von Benutzer erstellten Dateien und die installierte Software sowie das Betriebssystem selbst werden auf der microSD-Karte gespeichert.
- **USB-Tastatur und -Maus**, mit denen den Raspberry Pi gesteuert werden kann. Fast jede kabelgebundene oder kabellose Tastatur und Maus mit USB-Anschluss funktioniert mit dem Raspberry Pi.
- **Das HDMI-Kabel**, das Ton und Bilder vom Raspberry Pi auf Fernseher oder Monitor überträgt. Sie müssen nicht viel Geld für ein HDMI-Kabel ausgeben.

Die Arbeit mit einem RaspberryPi setzt ein paar Anfangsinvestitionen voraus, die auch von den angestrebten Aufgaben und Projekten abhängen. Zuerst gäbe es die Möglichkeiten, dass der gekaufte Raspberry Pi Board bereits ein Betriebssystem darauf installiert hätte. Aber es war nicht der Fall von vorhandenen im PSE-Labor Board. Um ein Betriebssystem auf diesen Raspberry Pi zu bringen, muss eine SD-Karte mit einem Betriebssystem-Image "geflasht" werden. Dafür zunächst wurde die Distribution von der Website Raspbian.org herunterladen und die MicroSD-Karte in den Kartenleser eines vorhandenen im PSE-Labor PC eingelegt. Anschließend wurde mit dem Macintosh Disk Utility-Dienstprogramm das heruntergeladene und entpackte Betriebssystem für den RaspberryPi auf eine Speicherkarte geschrieben. Dann ist die Karte in Raspberry Pi einzulegen. Der Raspi ist damit betriebsbereit und muss für die zukünftigen Anwendungen noch konfiguriert werden. Wenn der Pi zum ersten Mal eingeschaltet wird, wird viel Text auf dem Bildschirm angezeigt. Diese werden als Startmeldungen bezeichnet. Wenn Raspbian zum ersten Mal gestartet wurde, kann es ein oder zwei Minuten dauern, um die Nutzung des freien Speicherplatzes auf der microSD-Karte optimal anzupassen. Beim nächsten Start geht es schneller. Schließlich ist kurz ein Fenster mit dem Raspberry Pi-Logo zu sehen, dann wird Terminal Fenster angezeigt, in dem es einloggt werden muss. Zum

ersten Einloggen wird den Standardbenutzernamen "pi" und das Standardkennwort "raspberry" verwendet. Um Register-Client vor sowohl Online-Bedrohungen als auch von Missbrauch im Labor zu schützen, wurde das Standardkennwort sofort geändert. Der nächste Schritt ist Raspbian bis zur Version "Raspbian mit dem Raspberry Pi Desktop" zu aktualisieren, damit die grafische Benutzeroberfläche und Chromium Browser zu Verfügung stehen können. Dies kann mit dem Terminalbefehl gemacht werden:

```
sudo apt-get install lxde-core xserver-xorg xinit
```

Dann ist der Raspberry Pi erneut zu laden. Nachdem Raspberry Pi-Logo wieder angezeigt wurde, wäre der Raspbian-Desktop zu sehen. Somit gilt Betriebssystem als vollständig installiert und kann benutzt werden. Das war aber nicht der Fall mit dem vorhandenen Hardware, da es plötzlich eine Boot-Schleife vorkam, nachdem der Mikrocomputer eingeschaltet wurde und der Startvorgang nicht abgeschlossen werden konnte. Anstatt das zum Benutzung bereiteten Betriebssystem mit der grafische Benutzeroberfläche zu sehen, wird eine Schleife erzeugt, in der die Startvorgang kontinuierlich und wiederholt ausgeführt wurde und somit eine Nutzung der Mikrocomputers unmöglich ist. Nach den mehreren Recherchen wurde es vermutet, dass es durch eine unzureichende Stromversorgung verursacht werden könnte. Es wurde aber zuerst nicht versucht, einen USB-Netzteil zu wechseln, da die anderen USB-Netzteil man durch PSE-Labor bestellen und eine Zeit abwarten muss. Jedoch wurde eine erzeugte Boot-Schleife mit einem anderen Terminalbefehl erfolgreich gelöst:

```
sudo apt-get install --reinstall pcmanfm
```

Bei der Arbeit mit dem Mikrocomputer tritt jedoch später ein Problem mit der Stromversorgung auf. Der Fall kann im entsprechenden Kapitel 4.1.3 nachgelesen werden.

4.1.2 Installation der Treibers für RFID-Leser

Obwohl Raspbian mit einer Reihe von Software vorinstalliert ist, wird es aber zusätzlich benötigt, die Treiber für RFID-Leser zu installieren. Es sollte an dieser Stelle auch noch angemerkt werden, dass am Anfang der Entwicklungsprozess ein anderen RFID-Leser angeschlossen wurde als der, den in der Abschlussarbeit zu beschreiben und zu beobachten ist. Zuerst wurde die Treiber für Reiner SCT CyberJack RFID Basis[SCT] installiert. Die Schritte sollten in der Abschlussarbeit nicht unerwähnt bleiben, da die damals für den ersten RFID-Leser installierte Treiber und Daemons (Appendix 5.2) wurden endlich für den zweiten RFID-Leser benutzt, mit dem die Entwicklung der Aufgabe abgeschlossen wurde. Der Grund für die Hardwareaustausch ist die festgestellte Tatsache, dass Reiner SCT CyberJack RFID

Leser die Studentenkarten nicht ablesen konnte. Obwohl in der Spezifikation es steht, dass die Reiner RFID-Leser kontaktlose RFID Chipkarten wie eID mit dem neuen Personalausweis (nPA), GeldKarte oder eTicketing unterstützt, wurde es unmöglich mit den MIFARE-Transponder 13,561 MHz Funkbereich ins Spiel zu bringen. Die Studierenden-Ausweise der Beuth sollten mit dieser Technologie gelesen werden und somit ist diese Anforderung für die vorliegende Abschlussarbeit wichtig.

Um RFID-Leser anzuschließen, muss man den Leser über USB mit einem Raspberry Pi verbunden. Der Typ des RFID-Lesegeräts, der für die vorliegende Abschlussarbeit verwendet wird, ist ein ACR122U-A9 von Advanced Card Systems[Ltd], den auf der Abbildung8 zu sehen. Zuerst müssen wir die Paketlisten aktualisieren und einige Pakete herunterladen und installieren:[OG]. Die folgenden Abhängigkeiten werden benötigt im System mit dem Befehl "sudo apt-get install" zu installieren: *libusb-dev, libpcsc-lite-dev, libpcsc-lite1, libccid, pcscd, pcsc-tools, libpcsc-perl, libusb-1.0-0-dev, libtool, libssl-dev*.

PC/SC ist ein Standard für die Schnittstelle von Computern mit Smartcards, der auf den meisten Betriebssystemen, einschließlich Windows und Linux, verfügbar ist. PC/SC-Kopplungsgeräte benötigen einen Treiber, mit dem Anwendungen die Karte einfach erreichen können. Da PC/SC für Smartcards entwickelt wurde - und in einer Zeit, in der Smartcards nur Kontaktkarten waren funktioniert es auch mit den kontaktlosen Karten, falls RFID-Leser es unterstützt. Das Daemon-Programm für pcsc-lite namens pcscd koordiniert die Kommunikation mit Smartcard-Lesegeräten und Smartcards sowie kryptografischen Tokens, die mit dem System verbunden sind. Normalerweise wird pcscd beim Booten von /etc/init.d/pcscd gestartet.

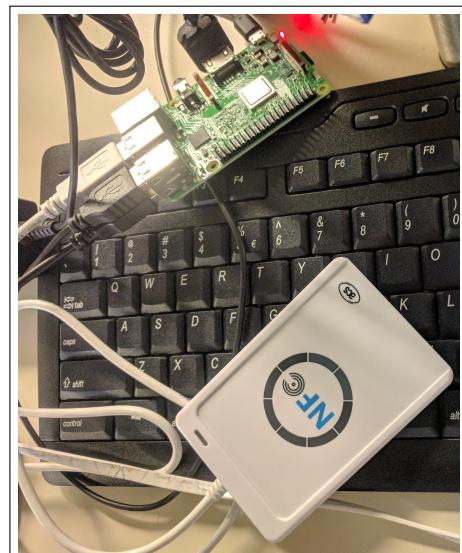


Abb. 8: ACR122U-A9 von Advanced Card Systems und Raspberry Pi

Damit können Anwendungen auf Smartcards und Lesegeräte zugreifen, ohne die Details der Karte oder des Lesegeräts zu kennen. Das Laden von Treibern für Kartenleser wird von pcscd koordiniert. Der Zweck von pcsc-lite besteht darin, eine kompatible API (Winscard) für die Migration von Windows-basierten PC / SC-Anwendungen auf Unix bereitzustellen[CR]. Die allgemeinen Zugriff auf USB-Geräte bietet eine C-Bibliothek namens libusb. Sie soll von Entwicklern verwendet werden, um die Produktion von Anwendungen zu erleichtern, die mit USB-Hardware kommunizieren. Es ist portabel, da mit einer einzigen plattformübergreifenden API auf USB-Geräte unter Linux, MacOS, Windows usw. zugegriffen werden kann. Sie wird im Benutzer-

modus ausgeführt und somit für die Kommunikation der Anwendung mit einem Gerät keine besonderen Berechtigungen oder Erhöhungen erforderlich sind.[lib]

Dann laden wir die Open-Source-Bibliothek libnfc für Near Field Communication (NFC) herunter, extrahieren, konfigurieren und installieren es. Nach der erfolgreichen Installation kann den verbindenden über USB RFID-Leser mithilfe des Tools lsusb angesehen werden: es wird VendorId und ProductID angezeigt. Es ist auch notwendig die abgelesene VendorId und ProductID in der entsprechenden XML-Datei. Die Datei auf der MicroSD-Karte ist zu finden :

```
/usr/lib/pcsc/drivers/ifd-ccid.bundle/Contents/info.plist
```

4.1.3 Spannungsproblem und Lösung

Nachdem der RFID-Leser angeschlossen wurde, tritt es ein weiteres Problem, das die freie Benutzung der Raspberry Pi unmöglich macht. Es wird unabhängig von der Zeit und vorherigen Geschehen eine Fehlermeldung "under voltage detected (0x000050000000)". Es wird zuerst versucht die kabellose USB-Tastatur und -Maus abzuschalten. USB eine aktive Abfrage seine Ports (Polling) benötigt, was bedeutet, dass die Anzahl der für andere Aufgaben verfügbaren CPU Zyklen geringer ist. Dies kann dazu führen, dass die CPU-Frequenz ansteigt, wodurch der Stromverbrauch höher wäre. Es wurde festgestellt, dass mit unverbundenen USB-Tastatur und -Maus die Fehlermeldung trotz vorkommt. Nur wenn der RFID-Leser angeschaltet wurde, kam es keine neue Fehlermeldung. Dann es wurde geprüft, ob ausgewählten RFID-Leser mit dem vorhandenen Mikrocomputer überhaupt kompatibel ist und nach der Lösung gesucht, mit der die zukünftige Entwicklung weiterlaufen kann.

Anfänglich wurde als Spannungsversorgungsteil ein USB-Netzteil von einem modernem bei Autorin der Abschlussarbeit vorhandenen zu Hause Smartphone benutzt. Das offizielle Raspberry Pi-Netzteil ist die empfohlene in der Dokumentation Wahl, jedoch wurde zuerst mit dem Mikrocomputer nicht gekauft. Ein leistungsfähiger USB-Netzteil könnte den schnell wechselnden Strombedarf des Raspberry Pi bewältigen. Nach der Besprechung des Problems mit den PSE-Labor Mitarbeiter wurde ein Samsung USB-Netzteil gegen einen Anker USB-Netzteil ausgetauscht und festgestellt, dass das Spannungsproblem mit den verbundenen sowohl USB-Tastatur und -Maus als auch RFID-Leser während des Ablesevorgangs nicht wieder erscheint.

4.1.4 Python Programmierung des RFID-Lesers

Nachdem es gelingt mir, die Hardware angefahren und die ersten Studentenkarte so abzulesen, dass es ein Schallton von RFID-Leser erzeugt wurde, sollte eine

weitere Aufgabe gelöst werden: die Daten von RFID-Tag auf einem auszuleihenden Raspi Board von Studentenkarte unterscheiden zu können. Es sollte ausgeschlossen werden, dass ein Ausleihe-/Rückgabevorgang angefangen wird, falls eine falsche Studentenkarte (z.B. mit einer BVG Jahresfahrkarte) am RFID-Leser präsentiert wird. Als es im Kapitel 2.3 erklärt wurde, sind die neuen Campus Karte der Beuth Hochschule für Technik Berlin mit den die MIFARE-Transponder hergestellt. Für die Programmierung des RFID-Lesers wird Smartcard-Schnittstelle benutzt, deren Installierung geschah zusammen mit pyscard und im Kapitel 4.1.2 geschrieben. Diese Schnittstelle steht für den Entwickler für die Arbeit mit Smartcards und NFC-Geräten zur Verfügung, sie wird in Form mehrerer Systemdienste implementiert und ihr Schnittstellenteil ist das PC/SC-Framework. PC/SC steht für Personal Computer/Smart Card.

Es steht mehreren Möglichkeiten für die Entwicklung eine Sprache zu wählen: die Funktionen der Smartcard-Schnittstelle können mit Python, C/C++ und Java Sprache verwendet werden. Die Programmierung des Register-Clients wird auf Python Sprache gemacht, dieselbe Sprache wird für Server während der Arbeit mit Django Framework benutzt. Die Entwicklung mit dem Importieren der PCSC-Header-Dateien beginnt.

```
from smartcard.System import readers
from smartcard.ATR import ATR
from smartcard.CardMonitoring import CardMonitor, CardObserver
from smartcard.CardRequest import CardRequest
```

Wann eine RFID-Leser über USB angeschlossen wird, kann es eine Verbindung zur PC/ SC-Bibliothek hergestellt und eine Liste der verfügbaren Terminals abgerufen werden. Alle API-Funktionen geben einen Statuscode zurück. Wenn die Funktion erfolgreich ist, wird die Konstante *SCARD_S_SUCCESS* zurückgegeben. Alle anderen Daten werden über Funktionsargumente zurückgegeben, in denen die Adresse der gewünschten Variablen übergeben wird. Die Verbindung (Initialisierung) erfolgt über die Funktion *SCardEstablishContext()*[Chi14, p. 101]. Die Adresse der Variablen *sc_context* vom Typ *SCARDCONTEXT* wird an sie übergeben. Als Nächstes müssen Sie eine Liste der Terminals abrufen. Dies erfolgt über die Funktion *SCardListReaders()*. Dann muss die Liste der angeschlossenen Lesers erhalten und gelesen. Es ist eine Reihe von Zeichenfolgen, die durch ein Nullbyte getrennt sind. Dies ist ein Windows-Format zur Darstellung von Zeichenfolgenlisten und wird üblicherweise als Zeichenfolge mit doppelter Nullterminierung bezeichnet. Durch Aufrufen der Funktion *SCardListReaders()* erhalten wir eine Liste aller Namen[Chi14, p. 102]. Da in Abschlussarbeit es ist vorgesehen, dass nur ein RFID-Leser über USB am Register-Client angeschlossen werden darf, wird nur ein Name nach der Funktionsaufruf zurückgegeben:

```
Found readers: ['ACS ACR122U']
```



Abb. 9: RFID-Leser mit der Studentenkarte der Autorin und RFID-Transponders

Wie die weitere Entwicklung darf einfach den ersten gefundenen RFID-Leser genommen werden. Wenn keine Terminals angeschlossen sind, wird das Programm beendet und eine entsprechende Meldung der PSE-Labor Mitarbeiter ausgegeben. Dann muss mit dem ausgewählten Terminal verbunden werden. Dies erfolgt durch Aufrufen der Funktion *SCardConnect()*. Beim Aufruf von *SCardConnect()* geben wir den gemeinsam genutzten Modus (*SHARE_SHARED*), das bevorzugte Kommunikationsprotokoll (in unserem Fall T0 und T1) an, übergeben die Adresse der Variablen, in die das Handle für weitere Operationen mit der Karte gespeichert wird, und die Adresse der Variablen *active_protocol*.

in der gewählten Kommunikationsprotokoll gespeichert wird.

Auf Anwendungsebene interessiert es uns nicht besonders, welches Protokoll zum Herstellen der Verbindung ausgewählt wurde, aber diese Informationen müssen während der Programmierung gespeichert werden - sie werden später zum Senden von Befehlen benötigt[Chi14, p. 104].

```
HRESULT hresult, hcontext = SCardEstablishContext(SCARD_SCOPE_USER)
if hresult == SCARD_S_SUCCESS:
    hresult, readers = SCardListReaders(hcontext, [])
    if len(readers) > 0:
        reader = readers[0]
        hresult, hcard, dwActiveProtocol =
            SCardConnect(hcontext, reader,
                        SCARD_SHARE_SHARED, SCARD_PROTOCOL_T0 |
                        SCARD_PROTOCOL_T1)
```

ATR (Answer-To-Reset) ist ein kurzes (nicht mehr als 33) Byte-Array, das die Karte beim Anschließen an das Terminal senden muss. Wenn die Karte dies nicht innerhalb einer bestimmten Zeit tut, wird davon ausgegangen, dass sie nicht richtig funktioniert. ATR enthält grundlegende Informationen über die Karte und die technischen Parameter der Verbindung. Das Format ist jedoch recht kompliziert und hängt vom Hersteller des Kartenchips, den darauf installierten „Anwendungen“ usw. ab. ATR wird im Terminal gespeichert, solange die Karte angeschlossen ist. Es lässt sich die zwei verschiedenen RFID-Transponder (MIFARE Smart-Studentenkarte und RFID-Tag) mit Hilfe ATR voneinander unterscheiden. Das folgendes wird z.B. von der Studentenkarte zurückbekommen, wann Autorin ihre eigene Studentenkarte am RFID-Leser ablesen lässt (siehe Abbildung 9):

```
+Inserted: 3B 81 80 01 80 80
Student card added
uid 04 2F 75 7A 30 40 80
```

Das andere wird aber angezeigt, wann den RFID-Transponder am RFID-Leser präsentiert wird, den in der Zukunft am im PSE-Labor vorhandenen Raspi-Boards angeklebt wird:

```
+Inserted: 3B8F8001804FOCA000000306030001000000006A
Raspi board added
uid 5C E7 87 30
```

Klasse *DetectionObserver* mit der Vererbung vom der Klasse *CardObserver* wird benutzt, um zu erkennen, wann eine Karte dem Kartenleser vorgelegt wurde, und dann die eindeutige Kennung (UID) von einer Karte zu lesen. Es wird mithilfe der Klasse *CardMonitor* getan, die das Einsetzen / Entfernen von Smartcards überwacht und den *CardObserver* benachrichtigt.

```
def update(self, observable, actions):
    (addedcards, removedcards) = actions
    for card in addedcards:
        atr = toHexString(card.atr)
        added_card = self.get_cardtype(toHexString(card.atr),
                                       "added")
        self.read_uid(added_card)
```

Sobald eine korrekte Studentenkarte oder RFID-Transponder am RFID-Leser erscheint und ohne Kollision mit anderen in der Nähe bleibenden elektromagnetische Felde abgelesen wird, wird abhängig vom dem Typ der Karte eine JSON-Datei erstellt und von einem acaLoan-client zum Server geschickt. Der acaLoan-client selbst ist ein kleinen Python-Script, der die korrekte Erstellung der JSON-Datei erlaubt und eine Verbindung mittels HTTP-Protokoll zum Server bedient.

```
class AcaLoanClient:
    def __init__(self, base_url):
        self.base_url = base_url

    def send_event(self, type, uid):
        endpoint = "{}/loan/api/events".format(self.base_url)
        payload = {"type": type, "uid": uid}
        r = requests.post(endpoint, json=payload)
```

Es wird weiteres von Register-Client auf keine Antwort vom Server erwartet, ob der abgelesenen Studentenkarte die Ausleihe des Raspi-Boards erlaubt ist oder ob gehaltenen in der Hand Raspi-Board ausgeliehen oder zurückgegeben darf. Sobald die Verbindung zum Server erfolgreich hergestellt wurde und eine JSON-Datei

geschickt, wird der RFID-Leser wieder zum Ablesen der nächsten RFID-Transponder freigegeben. Der Python-Script für RFID-Leser muss am Register-Client mit der Verwendung der Umgebungsvariable *LOANSERVER*. Während der Entwicklung der Abschlussarbeit geschieht es mit dem Name des Django Entwicklungsservers. Mehr darüber ist in folgenden Kapitel 4.2 nachzulesen.

```
LOAN_SERVER_URL=http://127.0.0.1:8000 python reader.py
```

Als letztes für die Implementierung des Register-Clients ist es wichtig, die kurze Beschreibung der entwickelten von Autorin JSON-Datei anzugeben. Definition der Struktur, des Inhalts und der Semantik von JSON-Objekten geschieht mithilfe von der Grammatiksprache namens JSON-Schema. Hier kann Metadaten (Daten über die Daten) angegeben werden, die die Eigenschaften eines Objekts beschreiben und gültige Werte.

```
{"$schema": "http://json-schema.org/draft-04/schema#",
"type": "array", "items": [
    {"type": "object", "properties": {
        "type": {"type": "string",
            "enum": ["card", "uid", "cancel_button", "terminate_button", "get_rfid_status",
                    "return_scanned_board_button", "loan_scanned_board_button", "finish_button"]},
        "uid": {"type": "string"}},
    "required": ["type", "uid"]}]}
```

Zu Zweck Definition der Struktur wird das Schlüsselwort "items" mit einem Array gesetzt, wobei jedes Element ein Schema ist, das jedem Index des Arrays des Dokuments entspricht. Das heißt, ein Array, bei dem das erste Element das erste Element des Eingabearrays validiert, das zweite Element das zweite Element des Eingabearrays validiert usw[MD]. Es ist zu betonen, dass für das ersten Element der JSON-Datei es eine Zeichenfolge aus einem festen Wertesatz sein muss. Nur diesen Zeichenfolgen können von Endliche Zustandsmaschine, die im Kapitels 3.6 und ?? beschrieben sind, abgearbeitet werden.

4.2 Server

Als er wurde im Kapitel 4.1.4 erwähnt, während der Implementierung der Aufgabe der Abschlussarbeit wurde mit dem Entwicklungsserver gearbeitet. Ein Entwicklungsserver ist ein Servertyp, der die Entwicklung und das Testen von Programmen, Websites, Software oder Anwendungen für Softwareprogrammierer erleichtert. Der bietet eine Laufzeitumgebung sowie alle Hardware- / Software-Dienstprogramme, die für das Debuggen und die Entwicklung von Programmen unerlässlich sind.

Django ist eines der effizientesten modernen Frameworks für die Entwicklung von Webprojekten. Der Grund für diese Effizienz ist ein klarer Mechanismus für die Arbeit mit einem Projekt, ein praktisches ORM (Object Relational Mapping Layer), mit der mit Anwendungsdaten aus verschiedenen relationalen Datenbanken wie SQLite, PostgreSQL und MySQL interagiert werden kann.

4.2.1 Erstellung acaLoan Django-Projekts

Glücklicherweise ist der Installationsprozess für Django unkompliziert, sodass das Einrichten Ihrer Entwicklungsumgebung schnell und entspannt ist. Django ist vollständig in Python geschrieben, daher muss zuerst Python installiert werden, um Django zu installieren. Weil die Autorin für die Implementierung des Servers ihren eigenen Mac OS Rechner verwendet, ist Python bereits Computer installiert. Wenn "Python" in die Befehlszeile eingegeben (mithilfe der Terminal.app auf meinem Mac), wurde folgendes angezeigt werden:

```
$ python
Python 3.8.1 (default, Feb 17 2020, 23:55:16)
[Clang 11.0.0 (clang-1100.0.33.17)] on darwin
```

Sobald Python auf Computer installiert ist, kann Django auch installiert werden. Es gibt drei Möglichkeiten: Installation der offizielle Django-Version, Verwendung eines verteilungsspezifisches Installationsprogramm oder Herunterladen der Version, die derzeit immer noch entwickelt wird. In der Abschlussarbeit wird nur die Installation der offiziellen Version verwendet. Der Installation wird mit "Pipenv"-Tool geschieht, das isolierte Python-Umgebungen bietet, die somit praktischer sind als die systemweite Installation von Paketen. Pipenv verwaltet Projektpakete automatisch über die Pipfile-Datei, während die Pakete installiert oder deinstalliert werden. Pipenv generiert auch die Datei Pipfile.lock, mit der deterministische Builds erstellt und eine Momentaufnahme Ihrer Arbeitsumgebung erstellt werden. Zusätzlich für die Aufgabeimplementierung wird django-fsm installiert, das mit der Django Installation nicht geliefert wird und erlaubt eine einfache deklarative Zustandsverwaltung für Django-Modelle.

Der ganzen Quellcode für das acaLoan-Server wird als Django Projekt dargestellt. Django hat einen Befehl zum einfachen Erstellen einer anfänglichen Projektstruktur. Es wird mit der Ausführung des folgenden Befehls vom Terminal erledigt:

```
django-admin startproject acaLoanRaspiBoard
```

Die Datei settings.py enthält eine Grundkonfiguration für die Verwendung einer SQLite-Datenbank und eine Liste von Django-Anwendungen und wurde als während

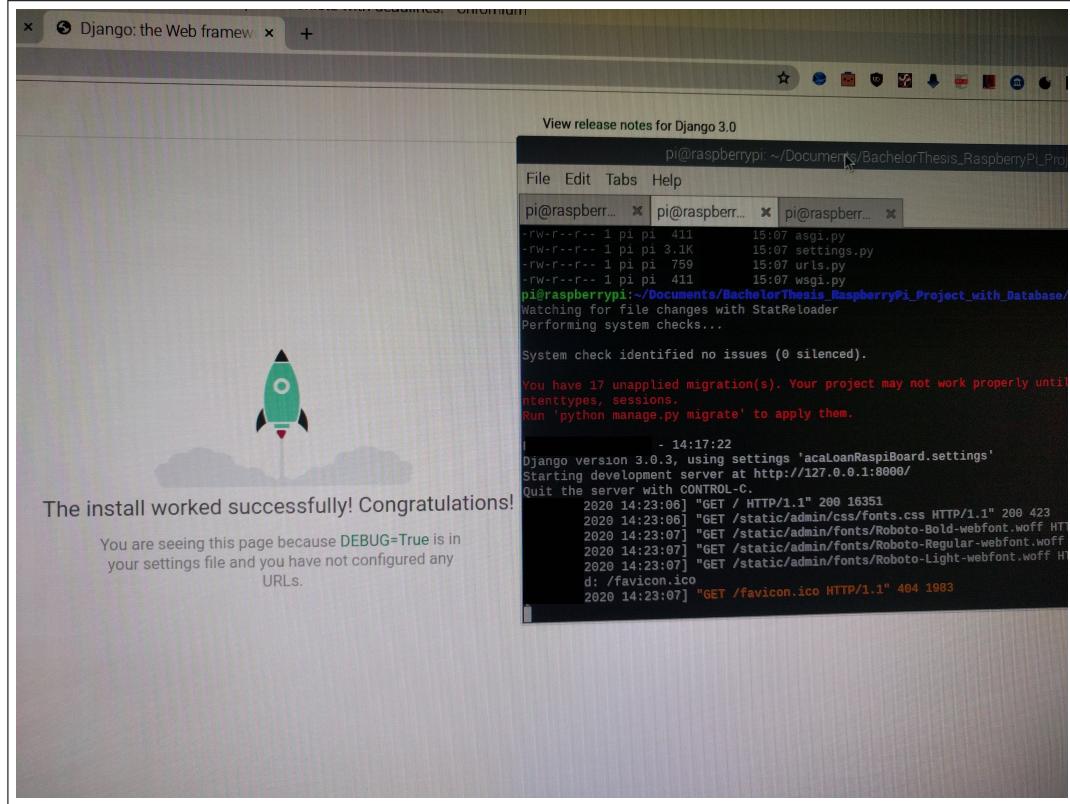


Abb. 10: Screenshot von Raspberry Pi nach dem Erstellung des acaLoanRaspiBoard-Projekts.

der Implementierung geändert, um das üblichste deutsche Zeitformat anstatt des amerikanischen zu verwenden.

```
LANGUAGE_CODE = 'en-us'
TIME_ZONE = 'Europe/Berlin'

TIME_INPUT_FORMATS = ['%H:%M:%S']
TIME_FORMAT = '%H:%M:%S'
DATETIME_FORMAT = "d.m.Y H:m"

DATE_FORMAT = "Y-m-d"
```

acaLoan-Server muss im zusätzliche Dateien wie Bilder, JavaScript oder CSS bereitstellen. In Django werden diese Dateien als "statische Dateien" bezeichnet. Django stellt django.contrib.staticfiles zur Verfügung, um den Entwickler bei der Verwaltung zu unterstützen. Dafür müssen in die Datei settings.py die folgenden Zeilen hinzugefügt werden:

```
STATIC_URL = '/static/'
STATICFILES_DIRS = [os.path.join(BASE_DIR, "staticfiles")]
```

In der settings.py ist auch den Debug-Modus des Projekts aktiviert, um während der Entwicklung detaillierte Fehlermeldungen zu sehen. Nach der Erstellung des

Projekts und kleine notwendigen Änderungen, kann endlich acaLoan-Server gestartet werden. Da Django einen praktischen Mechanismus bietet, um die Konfiguration Ihres Webservers während der Entwicklung zu vermeiden, kann enthaltenen bereits ein Webserver mit dem Befehl aufgerufen werden:

```
python manage.py runserver
```

Nach dem Befehl ausgeführt wurde und im Browser zu `http://127.0.0.1:8000/` navigiert wurde, wird die folgende Ausgabe angezeigt, die auf der Abbildung 10 zu sehen ist.

4.2.2 Datenbankeinrichtung

Nach dem Erstellung des Projekts und dem ersten Starten des acaLoan-Server ist es die Zeit, sich mit der Einrichtung von Datenbank zu beschäftigen. Theoretisch kann ein Django-basierte Projekt ohne Datenbank ausgeführt werden. Für die hier vorliegenden Abschlussarbeit was jedoch es ein Zweck, eine Datenbank einzurichten, um alle Ausleihe-/Rückgabevorgänge zu verwalten, so dass in einem echten Projekt kann man nicht auf Datenbankeinrichtung verzichten. Für die Implementierung wird für Verwendung von SQLite3 entschieden. Die Datenbank selbst wird in einer Datei auf der Festplatte gespeichert. Um die Datenbank mit dem Django-Projekt zu verbinden, muss den DATABASES-Block in der Datei `mysite / settings.py` bearbeitet. Um sqlite zu verwenden, muss nur `django.db.backends.sqlite3` als ENGINE angegeben und den Namen der Datei, in der die Datenbank gespeichert wird, in den Parameter NAME geschrieben werden[Foub].

Als nächsten müssen die Modelle in Django erzeugt werden, die die Daten definieren, mit denen während der Entwicklung gearbeitet werden. Die Modelle in Django werden entsprechend der Struktur realisiert, die im Kapitel 3.5 erwähnt wurde.

4.2.3 Erstellen der öffentlichen Schnittstelle - "Views"

4.2.4 Einführung in das Django Admin

4.2.5 Templates und Design der acaLoan-Website

4.2.6 Implementierung des Use Cases

Während der Implementierung der User Cases wurde auch Sequenzdiagramm erzeugt, die einfach die Interaktionen zwischen Objekten in einer sequentiellen

Reihenfolge zeigt, d.h. die Reihenfolge, in der diese Wechselwirkungen stattfinden. Die Sequenzdiagramm wurde als nächstes als eine Basis für das Design des Zustandsmaschine verwendet.

Sequenzdiagramm

Sequenzdiagramme beschreiben, wie und in welcher Reihenfolge die Objekte in einem System funktionieren. In der Abschlussarbeit werden ein Sequenzdiagramm gezeigt, die auf den Abbildungen 11 zu betrachten ist. Es ist ein erfolgreichen Szenario für die Ausleihe der Lab-Loan Raspi Board von einem gültigen Studierende. Das Sequenzdiagramme entspricht der oben genannte User Story "As a student, I want to loan a board so I can work at lab". Die meisten Kommunikation geschieht über die asynchrone Nachrichten. Zum Beispiel, es wird von Register-Klient mit dem angeschlossenen RFID-Leser nicht darauf gewartet, ob die abgelesene Studentenkarte gültig ist oder ob ein Student zum Kurs zugelassen ist. Sofort die vorherigen Ablesevorgang angeschlossen wurde und JSON-Datei dem Server geschickt, steht der RFID-Leser wieder zur Verfügung und ein RFID-Tag des Boards abgelesen werden kann. Es kann sogar sein, dass anstatt erwarteten in erfolgreichen Szenario RFID-Tags eine weitere Studentenkarte abgelesen wird (z.B. nebenstehende Kumpel der Studierende aus Spaß oder wegen der Eile lässt seine Karte ablesen vor dem Beenden des laufenden Ausleihevorgang). Obwohl es aus der Sicht der Zustandsmaschine nicht zugelassen ist, eine weitere Studentenkarte zu lesen, wird trotzdem die Karte von RFID-Reader abgelesen, eine weitere JSON-Datei zum Server geschickt und weiter RFID-Leser zum nächsten Lesen freigegeben. Es ist rein die Aufgabe des Servers die ankommenden Daten zu verifizieren und den Zustandsmaschine in einem weiteren Zustand zu schalten. Die entsprechende Information mit der Begrüßung der Studierende oder Fehlermeldung wird auch vom Server generiert und dem Display-Client zum Anzeigen übergeben. Mehr über diese Vorgehensweise in entsprechenden Kapitels 4.1.4, 4.2.6 und 4.3.1 der Implementierungsphase nachzulesen.

Endliche Zustandsmaschine

4.3 Display-Client

4.3.1 Clientseitiges JavaScript

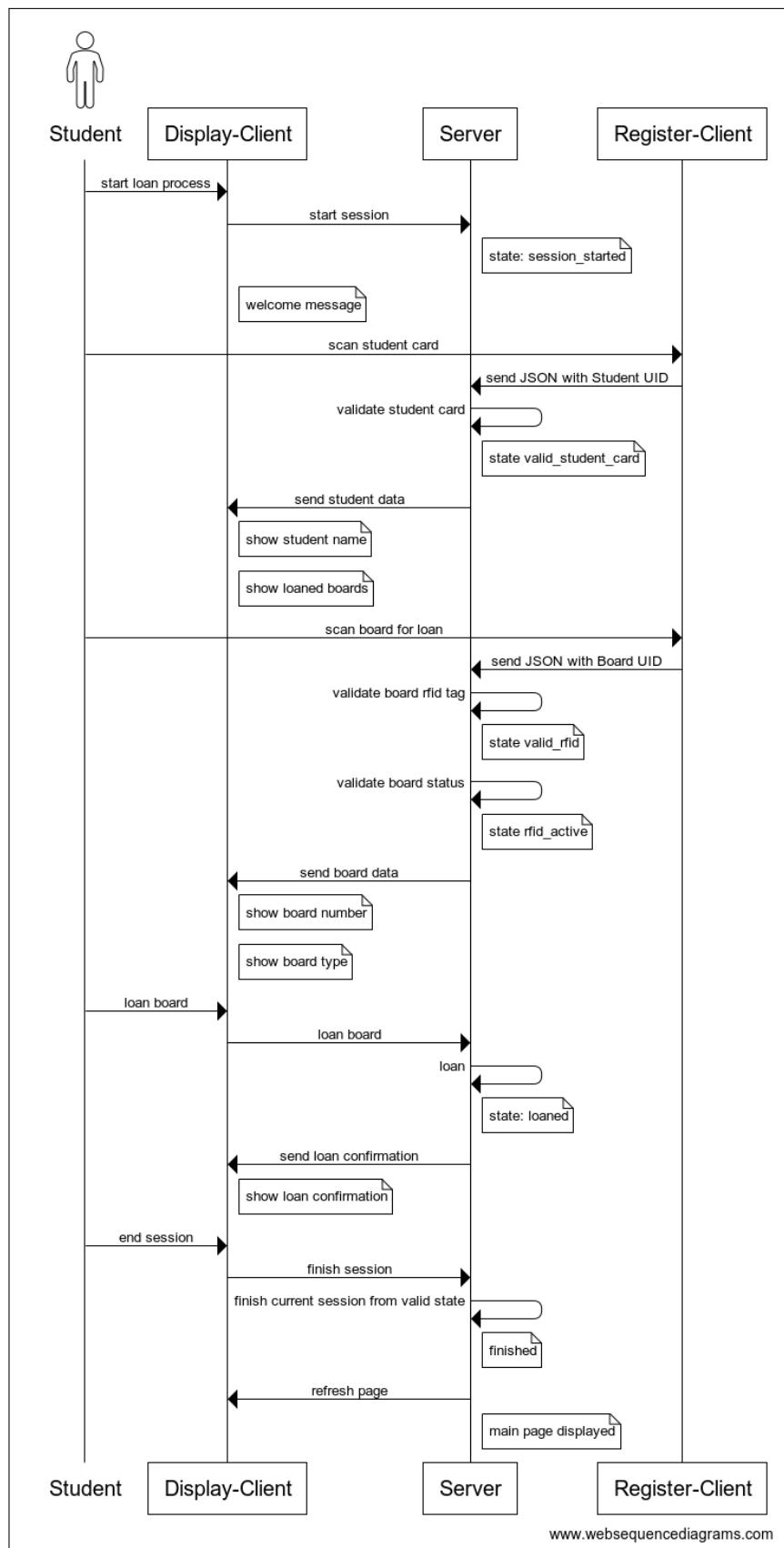


Abb. 11: Sequenzdiagramme der erfolgreichen Ausleihe der Lab-Loan Raspi Board

4.4 Automatisierten Tests

Results and conclusion

5.1 Results

5.2 Conclusion

Glossar

AJAX

AJAX (asynchrones Javascript und XML) ist der allgemeine Name für Technologien, mit denen asynchrone Anforderungen (ohne erneutes Laden von Seiten) an den Server gestellt und Daten ausgetauscht werden können. Da die Client- und Serverteile der Webanwendung in verschiedenen Programmiersprachen geschrieben sind, müssen zum Austausch von Informationen die Datenstrukturen (z. B. Listen und Wörterbücher), in denen sie gespeichert sind, in das JSON-Format konvertiert werden.

BGA

BGA oder Ball Grid Array ist eine Art oberflächenmontiertes Gehäuse, das in elektronischen Produkten zur Montage integrierter Schaltkreise wie Mikroprozessoren, FPGAs, WiFi-Chips usw. verwendet wird. Die Anschlüsse liegen in Form von Lötkugeln vor, die in einem Gitter angeordnet wie Muster auf der Unterseite des Gehäuses sind, um den für die Verbindungen verwendeten Bereich zu vergrößern.

Daemon

Ein Unix-Daemon ist ein Programm, das "im Hintergrund" ausgeführt wird, ohne dass die Steuerung über ein Terminal erforderlich ist, und dem Benutzer die Möglichkeit bietet, andere Prozesse "im Vordergrund" auszuführen. Der Dämon kann entweder von einem anderen Prozess gestartet werden, z. B. von einem der Systemstartskripte, ohne auf ein Steuerterminal zuzugreifen, oder vom Benutzer von einem beliebigen Terminal aus. In diesem Fall "entführt" der Dämon das Terminal jedoch nicht, während es ausgeführt wird.

DOM

DOM (Document Object Model) ist die Struktur einer HTML-Seite. Bei der Arbeit mit dem DOM werden HTML-Tags (Elemente auf einer Seite) gefunden, hinzugefügt, geändert, verschoben und entfernt.

GPU

Die Grafikverarbeitungseinheit ist ein programmierbarer Prozessor, der auf das Rendern aller Bilder auf dem Computerbildschirm spezialisiert ist. Eine GPU bietet die schnellste Grafikverarbeitung, und für Gamer ist die GPU eine eigenständige Karte, die an den PCI Express (PCIe) -Bus angeschlossen ist. GPUs wurden ursprünglich entwickelt, um Bilder für Computergrafik zu erstellen, jedoch seit Anfang 2010 können GPUs auch verwendet werden, um Berechnungen mit großen Datenmengen zu beschleunigen.

HOST

Host - Dies der Name der IP-Adresse für den Webserver, auf den zugegriffen wird. Dies ist normalerweise der Teil der URL, der unmittelbar auf den Doppelpunkt und zwei Schrägstriche folgt.[RR03, p.31]

HTML-Vorlage

Eine HTML-Vorlage ist eine intelligente HTML-Seite, die Variablen anstelle bestimmter Werte verwendet und verschiedene Operatoren bereitstellt: if (if-then), for-Schleife (Durchlaufen einer Liste) und andere. Das Abrufen einer HTML-Seite aus einer Vorlage durch Ersetzen von Variablen und Anwenden von Operatoren wird als Vorlagenrendering bezeichnet. Die resultierende Seite wird dem Benutzer angezeigt. Falls einen anderen Abschnitt zu öffnen ist, muss ein anderen Musters geladen werden. Wenn andere Daten in der Vorlage verwendet werden müssen, werden sie vom Server angefordert. Alle Formularübermittlungen mit Daten sind auch AJAX-Anforderungen an den Server.

HTTP

HTTP steht für HyperText Transfer Protocol, Hypertext Transfer Protocol". HTTP ist ein weit verbreitetes Datenübertragungsprotokoll, das ursprünglich für die Übertragung von Hypertextdokumenten vorgesehen war (Dokumente, die möglicherweise Links enthalten, mit denen Sie den Übergang zu anderen Dokumenten organisieren können). Die Basis dieses Protokolls ist eine Anforderung von einem Client (Browser) an einen Server und eine Serverantwort an einen Client.

JSON

JSON (JavaScript Object Notation) ist ein universelles Format für den Datenaustausch zwischen einem Client und einem Server. Es ist eine einfache Zeichenfolge, die in jeder Programmiersprache verwendet werden kann.

PORT

Dies ist ein optionaler Teil der URL, der die Portnummer angibt, die der Zielwebserver abhört. Die Standardportnummer für HTTP-Server ist 80, einige Konfigurationen sind jedoch so eingerichtet, dass sie eine alternative Portnummer verwenden. In diesem Fall muss diese Nummer in der URL angegeben werden. Die Portnummer wird direkt mit einem Doppelpunkt, der unmittelbar auf den Servernamen oder die Adresse folgt, eingegeben.[RR03, p.31]

RFID

RFID (englisch radio-frequency identification oder „Identifizierung mit Hilfe elektromagnetischer Wellen“) bezeichnet eine Technologie für Sender-Empfänger-Systeme und wird bei der Abschlussarbeit verwendet, um die vorhandenen zur Ausleihe Raspberry Pi Boards zu markieren und identifizieren.

URI

Uniform Resource Identifier ist ein Pfad zu einer bestimmten Ressource (z.B. einem Dokument), für die eine Operation ausgeführt werden muss (z. B. bei Verwendung der GET-Methode bedeutet dies das Abrufen einer Ressource). Einige Anforderungen

beziehen sich möglicherweise nicht auf eine Ressource und in diesem Fall kann der Startzeile anstelle des URI ein Sternchen (Symbol "") hinzugefügt werden.

Literaturverzeichnis

- [Ado] Adobe. *Understand web applications*. <https://helpx.adobe.com/dreamweaver/user-guide.html/dreamweaver/using/web-applications.html>. abgerufen am 29. September 2020.
- [Chi14] Ugo Chirico. *Smart Card Programming. A comprehensive guide to smart card programming*. lulu.com London, 2014. ISBN: 978-1-291-61050-5.
- [CR] David Corcoran and Ludovic Rousseau. *pcscd(8) - Linux man page*. <https://linux.die.net/man/8/pcscd>. abgerufen am 05. Oktober 2020.
- [Foua] Django Software Foundation. *Writing your first Django app, part 1*. <https://docs.djangoproject.com/en/3.1/intro/tutorial01>. abgerufen am 12. Oktober 2020.
- [Foub] Django Software Foundation. *Writing your first Django app, part 2*. <https://docs.djangoproject.com/en/3.1/intro/tutorial02>. abgerufen am 13. Oktober 2020.
- [Geo17] Nigel George. *Mastering Django: Core. The Complete Guide to Django 1.8 LTS*. Leanpub book, 2017. ISBN: 978-0-9946168-0-7.
- [Hal19] Gareth Halfacree. *The Official Raspberry Pi. How to use your new computer. Beginner's Guide*. aspberry Pi Press; 3rd Revised edition, 2019. ISBN: 1912047586.
- [JI] Ralph Jacobi and Josh Wyatt. Texas Instruments. *MIFAREDESFireEV1 AES Authentication WithTRF7970A*. <https://www.ti.com/lit/an/sloa213/sloa213.pdf>. abgerufen am 07. Oktober 2020.
- [lib] libusb. *libusb-1.0 API Reference. A cross-platform user library to access USB devices*. <http://libusb.sourceforge.net/api-1.0>. abgerufen am 05. Oktober 2020.
- [Ltd] Advanced Card Systems Ltd. *ACR122U USB NFC Reader*. <https://www.acs.com.hk/en/products/3/acr122u-usb-nfc-reader/>. abgerufen am 05. Oktober 2020.

- [MD] Space Telescope Science Institute. Michael Droettboom. *Understanding JSON Schema*. <https://json-schema.org/understanding-json-schema/reference/array.html>. abgerufen am 10. Oktober 2020.
- [NXP] NXP. *MIFARE® contactless tag IC family overview*. <https://www.nxp.com/docs/en/product-selector-guide/MIFARE-ICs-linecard.pdf>. abgerufen am 06. Oktober 2020.
- [OG] One Blog: it is all in the name. One Guy. *ACR122U NFC USB READER ON A RASPBERRY PI*. <https://oneguyoneblog.com/2015/09/16/acr122u-nfc-usb-reader-raspberry-pi/>. abgerufen am 05. Oktober 2020.
- [Pri] Author: SV 2d/3d Printing. *Raspberry Pi programming basics*. <https://api-2d3d-cad.com/microcomputer-programming-basics/>. abgerufen am 03. Oktober 2020.
- [RR03] Leon Shklar und Richard Rosen. *Web Application Architecture. Principles, protocols and practices*. John Wiley & Sons Ltd, 2003. ISBN: 0-471-48656-6.
- [SCT] Reiner SCT. *cyberJack® RFID basis. Chipkartenleser für den neuen Personalausweis*. <https://shop.reiner-sct.com/chipkartenleser-fuer-die-sicherheitsklasse-3/cyberjack-rfid-basis>. abgerufen am 04. Oktober 2020.
- [TBa] Beuth Hochschule für Technik Berlin. *acaLab: Aktivitäten und Abschlussarbeiten*. <https://labor.beuth-hochschule.de/pse/acalab-aktivitaeten-und-abschlussarbeiten>. abgerufen am 21. August 2020.
- [TBb] Beuth Hochschule für Technik Berlin. *Campus-Card*. <https://www.beuth-hochschule.de/campus-card>. abgerufen am 08. Oktober 2020.
- [TBC] PSE-Labor der Beuth Hochschule für Technik Berlin. *acaLoan-Raspi*. <https://labor.beuth-hochschule.de/pse/acalab-aktivitaeten-und-abschlussarbeiten/acaloan-raspi/>. abgerufen am 15. Oktober 2020.
- [Tea] Unito Team. *3 Easy Ways to Add Agile Methodology to Your GitHub Projects While Keeping Developers Happy*. <https://unite.io/blog/github-projects-agile/>. abgerufen am 16. Oktober 2020.
- [Tec] Syrma Technology. *How Radio-Frequency Identification Systems Work*. <https://www.syrmatech.com/rfid-system/>. abgerufen am 09. Oktober 2020.
- [Too] NFC Tools. *ISO/IEC 14443 Type A*. <http://nfc-tools.org/index.php/ISO14443A>. abgerufen am 07. Oktober 2020.