|  |  |  |
| --- | --- | --- |
| **Kurs:** Distributed & Mobile Systems | | **Code:** W6DIST-20 |
| **Studiengang** | BSc Wirtschaftsinformatik | |
| **Studiengruppe** | BWI-A20 | |
| **Dozent/in** | Walter Rothlin / Frank-Stefan Heinz | |
| **Leistungsnachweis(e)** | Einzelarbeit mit Fachgespräch | |
| **Präsenz** | **Selbststudium** | **Workload** |
| 32 h | 40 h | 72 h |

|  |
| --- |
| **Lernergebnisse** |

Die Studierenden

* Sie können eine Client-Server Applikation auf der Basis von TCP/IP Sockets realisieren.
* Sie können verteilte Applikationen konzipieren und entwickeln.
* Sie kennen die grundlegenden Konzepte, Techniken und Methoden, um verteilte und mobile Softwaresysteme zu designen und in Python zu implementieren.
* Sie sind in der Lage, bei Software-Architekturfragen einen entscheidenden Beitrag zur Lösung beitragen.
* Sie kennen die wichtigsten SOA – Konzepte (SOAP / REST) und können einfache Webservices selbst in Python entwickeln

|  |
| --- |
| **Lehrinhalte** |

1. TCP/IP Socket communication
2. XML-Processing in Python
3. Services via TCP/IP (csv und xml)
4. 2-Tier Application (Python Frontend / DB as Backend)
5. Bestehender Web-Services (SOAP / REST) in Python nutzen
6. Message queues in Python
7. Mobile Entwicklung und Marketing Rundgang (Frank-Stefan Heinz)
8. Eigener Web-Service in Python implementieren
9. IoT Anwendung entwickeln (Compose services to a new service)

|  |
| --- |
| **Lehr- und Lernmethoden** |

* Stoffvermittlung Frontalunterricht, Übungen lösen im Präsenzunterricht und Selbststudium
* **Formative Lernkontrolle:** Zur Vorbereitung auf den Leistungsnachweis müssen die Studierenden zu Fragestellungen entsprechende Lösungen in Python implementieren und die Resultate selbständig überprüfen. Als Entwicklungs-Umgebung wird PyCharm verwendet.

|  |
| --- |
| **Leistungsnachweis** |

Eine Applikation anhand von vorgegebenen Requirements nach den Clean-Code Regeln in Einzelarbeit vor Ort entwickeln. Es sind alle Unterlagen, Beispielcode oder Google erlaubt. Es muss eine Ehrenwörtliche Erklärung abgegeben werden, dass die Arbeit alleine ohne fremde Hilfe erstellt wurde.

Geprüft wird der Code anhand der funktionalen und Qualitäts-Kriterien, welche im abgegebenen Prüfungsdokument festgelegt werden. Anhand eines Fachgespräches wird überprüft, ob der Student den Code und die darunter liegenden theoretischen Grundlagen verstanden hat. (Gewicht: 100%)

|  |
| --- |
| **Lehrmittel** |

Verpflichtend

Inhalt vom Module Web- & N-tier Applications verstanden. Python-Kurse bestanden und beherrscht seine Entwicklungsumgebung.

|  |
| --- |
| **Semesterprogramm** |

|  |  |
| --- | --- |
| 1. **Client-Server Kommuniation** | |
| Datum | Do 23.3.2023 |
| Zeit | 17:30 – 20:45 |
| Dozent | Walter Rothlin |

Vorbereitung

* OSI Schichten Modell
* XML und XML Processing
* Python String-Methoden (split, join,…) anwenden können

Inhalte

* Ein eigener Client und Server in Python implementieren, welcher über eine Socket-Communication Daten austauscht.
* Die drei Grundoperationen (Validierung, Transformation und Data-Access) mit einem XML-Dokument in Python implementieren
* Erweitern der TCP/IP Socket Applikation als Service Call

Nachbereitung

* Testen und Fertigstellen der Applikation

|  |  |
| --- | --- |
| 1. **Verschlüsselung / Authentification** | |
| Datum | Do 13.4.2023 |
| Zeit | 13:15 – 16:30 |
| Dozent | Walter Rothlin |

Vorbereitung

Inhalte

Nachbearbeitung

|  |  |
| --- | --- |
| 1. **2-Tier Application (Python Frontend / DB as Backend)** | |
| Datum | Do 20.4.2023 |
| Zeit | 17:30 – 20:45 |
| Dozent | Walter Rothlin |

Vorbereitung

* DB-Anbindung in Python (Siehe Database Module)

Inhalte

* DB-Connection in Python
* SQL-Abfragen (auf Tabellen, Views, Meta-Daten inkl. Functions)
* Calling a Stored-Procedure
* Processing Result-Sets

Nachbearbeitung

* GUI-Application für Adress-DB entwickeln

|  |  |
| --- | --- |
| 1. **Server-seitiges programmieren: CGI-WSGI / FLASK-DJANGO** | |
| Datum | Do 11.5.2023 |
| Zeit | 17:30 – 20:45 |
| Dozent | Walter Rothlin |

Vorbereitung:

* Request / response
* http-Round trip
* Parameterübergabe (Put/Get)

Inhalt:

* Common Gateway Interfaces / Web Server Gateway Interface
* FLASK / Django

Nachbearbeitung:

* Eigener REST-Service in FLASK implementieren und in CLI-Applikation nutzen

|  |  |
| --- | --- |
| 1. **Leistungsnachweis** | |
| Datum | Do 1.6.2023 |
| Zeit | 17:30 – 20:45 |
| Dozent | Walter Rothlin |

Inhalte

* **Abgabe der Aufgabenstellung und Kriterien für den Leistungsnachweis**

|  |  |
| --- | --- |
| 1. **Message Queues in Python** | |
| Datum | Do 15.6.2023 |
| Zeit | 17:30 – 20:45 |
| Dozent | Walter Rothlin |

Vorbereitung

Funktion und Aufgabe eines Message Broker erklären können

Inhalte

* Message oriented programing in Python mit RabbitMQ
* **Parallel dazu Fachgespräche zur eigenen Arbeit (pro Person max 10‘)**

Nachbereitung

* Applikation testen

|  |  |
| --- | --- |
| 1. **Mobile Entwicklung und Marketing ein Rundgang** | |
| Datum | Do 29.6.2023 |
| Zeit | 17:30 – 20:45 |
| Dozent | Frank-Stefan Heinz |

Inhalte

* Mobile Entwicklung und Marketing ein Rundgang

|  |  |
| --- | --- |
| 1. **Web-Services Orchestration (Compose services to a new service)** | |
| Datum | Do 13.7.2023 |
| Zeit | 17:30 – 20:45 |
| Dozent | Walter Rothlin |

Inhalte

* IoT Anwendung selber entwickeln (inkl. QR-Code)
* Beispiel Smart Home
* **Parallel dazu Fachgespräche zur eigenen Arbeit (pro Person max 10‘)**

|  |  |
| --- | --- |
| 1. **IoT Anwendung entwickeln (Compose services to a new service)** | |
| Datum | Do 13.7.2023 |
| Zeit | 17:30 – 20:45 |
| Dozent | Walter Rothlin |

Inhalte

* IoT Anwendung selber entwickeln (inkl. QR-Code)
* Beispiel Smart Home
* **Parallel dazu Fachgespräche zur eigenen Arbeit (pro Person max 10‘)**

|  |  |
| --- | --- |
| 1. **Bestehender Web-Services (SOAP / REST) in Python nutzen** | |
| Datum | Do 20.4.2023 |
| Zeit | 17:30 – 20:45 |
| Dozent | Walter Rothlin |

Vorbereitung

* Web-Services (Unterschied SOAP / REST) konzeptionell verstehen

Inhalte

* Web-Service WS in Java: Konzept verstehen und von Python aus nutzen
* API eines public Web-Service (Google, Open-Weather, Map.geo.admin,…) verstehen
* Applikation in Python implementieren, welche einen *open* WEB-Service nutzt

Nachbereitung

* Applikation testen und fertig implementieren