This WASA course unit [CM-W-INT] describes the content and the organization of the lecture and practical course "Web Applications and Service-oriented Architecture" (WASA) provided by the research group Cooperation & Management (C&M, Prof. Abeck). Current concepts of software development and architectures (including Domain-Driven Design, Behavior-Driven Development, Microservices, RESTful Webservices) as well as related standards and technologies (including HTTP, JavaScript/TypeScript, Angular, Spring) are introduced which are needed to develop advanced mobile web applications. The web applications stem from the connected car domain which includes concepts from the domain of Internet of Things. Since the concepts presented in the lecture must be practically applied to really understand them the WASA lecture is only offered in a combination with the WASA practical and/or seminar course.

Further remarks:
(1) The lecture takes place each **Wednesday from 9:45 am to 11:15 am** in the computer science building "Am Fasanengarten" (building number **50.34**) in the room **SR301**.
(2) The written lecture content is made available in **English**. In the lecture, the content is presented and discussed in **German**. Each student participating in the WASA lecture can choose if he/she wants to write his/her practical/seminar thesis in **German** or in **English**. Thesis templates are made available both in LaTeX and in Word.

C&M Cooperation & Management
HTTP HyperText Transfer Protocol
KIT Karlsruhe Institute of Technology
REST REpresentational State Transfer
WASA Web Applications and Service-oriented Architectures

The research work carried out by C&M can be divided into two main areas:

(Connected Car, Internet of Things) In this area connected car application and IoT application based on the concept of domain modeling and microservice architectures are developed. Relevant concepts applied in the development process include Behavior-Driven Development (BDD) and Domain-Driven Design (DDD).

(Identity and Access Management, SecDevOps) SecDevOps concerns the continuous integration (CI) and continuous deployment (CD) of (hopefully not monolithic, but microservice-based) software systems deployment into a container-virtualized (Docker/Kubernetes-based) cloud infrastructure. The Identity and Access Management (IAM) is a highly relevant crosscutting concern appearing in almost every web application. One of the projects is concerned with the provisioning of IAM as a flexible cloud service.

BDD       Behavior-Driven Development
CI/CD     Continuous Integration / Continuous Deployment
DDD       Domain-Driven Design
IAM       Identity and Access Management
IOSB      Fraunhofer Institute of Optronics, System Technologies, and Image Exploitation
          (Optronik, Systemtechnik und Bildauswertung)
IoT       Internet of Things
SecDevOps Security Development Operations
The software development process applied by C&M combines the concepts of Behavior-Driven Development (BDD) and Domain-Driven Design (DDD). Both concepts provide complementary contributions to the layered microservice architecture as the figure illustrates.

(1) An implemented feature can be seen as a deployable increment of the software system. (Feature 1, Feature 2, ...) The ordering of the features in the figure implies that the first feature should cover the core functionality of the software system.

(2) The domain model makes sure that the static and dynamic domain knowledge is consistently used by each feature. This ensures that the features build a consistent whole although each feature is developed and deployed independently from other features.
The activity diagram gives an overview of the whole engineering process which in fact is divided into two sub-processes: According to our understanding of DDD, a domain model is the foundation of all software applications belonging to this domain. We regard the domain model as the knowledge of a domain expert he or she makes available to the software developer by the model. This knowledge is the application-agnostic part of the application meaning that each application of the domain implements this knowledge in its domain logic layer. We understand this domain knowledge as the "physics" that each application must adhere to. Evans calls this part of functionality the heart of the software system [Ev03].

(Starting Points) (Strategic Modeling of the Initial Context Map) The starting point of the domain modeling process is set earlier as the starting point of the application development process. The reason is that it makes sense to have an initial context map of the domain before the development of the first application starts. In the following we describe the main characteristics of the application development process.

(Context Map) The main artifact of the domain modeling process is the context map in which all micro-service-based applications belonging to the domain must fit into.

(Strategic Modeling of the Bounded Contexts as Part of the Domain's Context Map) In the application development process, the integration of the application under development into the context map of the domain is done in the second step after the first step in which the BDD-based requirements analysis is carried out.

(Tactical Modeling of a Bounded Context) A bounded context is part of a context map. It represents a candidate for a microservice which can be developed by an independent team [Ne15]. DDD considers the modeling of the content (i.e. the functionality) of a bounded context as tactical modeling.

(Relation View) The most relevant artifact of the tactical modeling is the relation view which is built in the third step. By the relation view central DDD concepts (such as entity and value objects) including their relationships and operations are represented.

(Implementation of each Bounded Context as a Microservice Accessible via its Web API) The functionality modeled by the relation view is implemented in the domain logic layer of the microservice architecture. A systematic approach to implement the bounded context as a microservice (fourth step) is a core characteristic of the proposed development process. One of the key demands of DDD is to keep the model and its implementation in sync in order to assure that the model and the implementation is the same.

The two remaining steps concern development activities around the core implementation of the microservice, namely the implementation and test and the deployment of the application.

The main dimension according which the development tools can be ordered are the development phases (from analysis to deployment). The tools supporting the analysis and design and the implementation and testing can be grouped according to the software architecture (which in our case is a microservice architecture).

(Project Management and Version Control) These tools support the overall organization of the software project and the communication between the project members. At C&M two different tools sets are used: the Atlassian toolset extended by Microsoft tools, esp. SharePoint on which the C&M Teamserver is based.

(Analysis and Design) Analysis requirements at C&M is done by taking the approach of behavior-driven development (BDD) based on the tool Cucumber. For the design the two most relevant tools are Enterprise Architect for the domain model and Swagger for the API specification. In addition to these tools the Microsoft Office tools (Word, PowerPoint) and Atlassian Confluence (in the case of iCC/xdi projects) are applied for documentation purposes.

(Implementation and Testing) In this phase the frontend and backend of the web applications are constructed. The integrated development environment (IDE) used for frontend development is JetBrains's WebStorm and frontend frameworks are Angular and Bootstrap. As IDE for backend development Eclipse is used and Apache Spring (esp. Spring Boot for the microservice implementation) is used as backend framework.

(Build and Deployment) The build and deployment of the microservices is carried out via a build pipeline by which the concept of continuous integration and continuous deployment is provided. The result of the build pipeline is a Docker image (= application container image) since Docker is used as the container environment at C&M. Kubernetes is used to manage the Docker containers in order to reach a high scalability and robustness of the service landscape.

BDD Behavior-Driven Design
IDE Integrated Development Environment
The table describes the planned sequence of course units presented during the lecture.

(Content) All titles printed in upper case are course units that are the basis for the oral examination carried out at the end of the semester. After the course unit was held a range of page numbers are added. This indicates which pages must be prepared by the student for the oral examination.
The acronym WASA stands for "Web Applications and Service-orientierted Architectures". Four different types of WASA courses are offered: (i) lecture courses WASA1 and WASA2 (ii) practical courses WASA1 and WASA2 associated to the lecture courses (iii) proseminar course associated to WASA1 lecture course and seminar associated to WASA2 lecture course (iv) key qualification course (germ. Schlüsselqualifikation SQ).

1. The lecture courses WASA1 and WASA2 each comprise 2 semester hours. A student who attends one of the lectures acquires 4 credit points (germ. Leistungspunkt).

2. The practical course runs in parallel with the lecture course and counts 5 credit points meaning a workload of 150 hours. The capacity of students C&M can offer the practical course depends on the current projects carried out in the research group.

Hint: In the Wirtschaftsinformatik study programme the name of the module is "Microservice-basierte Web-Anwendungen".

3. The duration of the oral exam is 20 minutes. The examiners are Prof. Abeck and one of the C&M's PhDResearchers. Since the examination is in the lecture term the students should have a good personal resource management in order to have enough time for the preparation of the examination.

SQ Schlüsselqualifikation (Key Qualification)
On this page the specifics of the WASA practical/seminar courses which are offered in parallel to the WASA lecture are described.

(1) A project consists of about 4 to 6 practical/seminar students.
(1.1) A SeniorStudent is a student who is writing his/her bachelor thesis or master thesis at C&M.
(1.2) The meetings take place at a defined time which is fixed at the beginning of the semester.

(2) The topics dynamically evolve from the work done by the SeniorStudent in their bachelor/master thesis.
(2.1) The JuniorStudent should actively participate in the discussion and make own proposals how the topic should be treated.
(2.2) Reviews are an integral part of the work in the project team.
(2.3) This means that the practical/seminar work has a dynamic characteristic.
(2.4) There is a high flexibility and liberty concerning the focal points of the practical/seminar work.

(3) During the first phase of the practical/seminar course all JuniorStudents fulfill the same tasks. After the successful completion of this phase each JuniorStudent knows how to use the major tools (esp. GitLab) to build microservice-based applications.
The task document (Aufgabendokument) is a Word document which describes the tasks a JuniorStudent has to fulfill during his/her practical/course. The document is stored in the personal working folder on the C&M Teamserver (further described on a following page <<C&M Teamserver>>).

1.1 The single tasks to be carried out and the results to be achieved during the initial phase can be found in the task document.

1.2 These tasks are specified individually for each JuniorStudent by the co-coaching PhDResearcher or SeniorStudent.

2. The exact delivery date can be found in the task document.

2.1 The result of the review is available one week after the delivery of the document (stored in the review folder of the JuniorStudent).

3. There are templates available for the different variants in which a JuniorStudent can write his/her practical/seminar thesis.
The figure illustrates which research areas are covered by the Bachelor/Master/PhD theses currently worked out at C&M. Each student who takes part in the WASA lecture will be assigned as a JuniorStudent to one of these theses and the related SeniorStudent/PhDResearcher. The SeniorStudent/PhDResearcher is the co-supervisor of the practical/seminar work carried out by the JuniorStudent. A SeniorStudent/PhDResearcher and the JuniorStudents assigned to him/her make up a project team (PT). The PT meetings take place in the Team Room 032 (alternatively in the NZ1 Seminar Room in the case that Room 032 is occupied).

In the following, each PT is presented by introducing into the topic and by showing how the JuniorStudents will be integrated into the work of the SeniorStudent/PhDResearcher who is leading the PT (Schneider, Holzmann, Gassauer, Kostka; Hippchen; Sidler; Hoyer; Throner, Braun, Qattan)

<table>
<thead>
<tr>
<th>PT</th>
<th>Project Team</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The goal is to create a clearly structured microservice architecture for IoT-based applications. This means that no business parts are integrated into the IoT domain and vice versa. In order to provide a microservice architecture for the IoT domain, it needs to be divided into several functional services. The Domain-Driven Design (DDD) approach is used in order to determine the domain logic of the services.

1. A structured development process for IoT applications is established by extending the currently applied C&M development process. The extended process leads to additional artifacts in order to handle the IoT aspects of an application.

2. Different artifacts, such as the context map and an application-driven development process are already established. However, the applicability of the artifacts has to be shown. Therefore, one goal of this project team is to verify if the established artifacts are useful and at which points adjustments (or additional artifacts) are required.

The following points show the work packages of the IoT project team.

1. One of the first goals is to get a brief understanding of the different standards, such as OGCs SensorThings API [OGC-Sen] and W3Cs Web of Things [W3C-Web] in order to get a basic understanding of IoT problems and proposed solutions.

2. When one applies the development process, the necessary artifacts are created. One step of the development process applies a filter between the applications business domain and the IoT specific domain. The goal behind this step is to separate the IoT domain and the business domain. This process should be applied to an existing IoT application (like ECC) and as a result, it should reveal the architectural deficiencies of the application.

3. The extended development process is applied to one of the application of the connected car domain. The goal is to separate the previous mentioned IoT and the business specifics.

4. By the use of the previous generated artifacts, a prototype of an IoT microservice should be developed.

5. This work package refers to the previous ones and is intended to reveal problems and missing artifacts in the application of development for IoT applications.

ECC Electric Car Charger
OGC Open Geospatial Consortium
W3C World Wide Web Consortium

Various sensors (oil pressure, GPS data, brake sensor data) from a car collect important (live) data which is valuable for the car manufacturer. Those sensor data should be transmitted as one or multiple streams.

(1.1) With the usage of policies the car manufacturer should be able do adjust which sensor data is transmitted into the cloud.
(1.1) Existing applications (CarSharing application) and the connected car domain must be considered, because it should be examined how streaming service can be included into these applications [CM-W-CON].

(2) The following points illustrate the work packages for JuniorStudents.
(2.1) Modern cars are IoT devices, therefore IoT standards like the (SensorThings API) are important [OGC-Sen] [CM-W-CON].
(2.2) The C&M research group already developed microservice based applications within the connected car domain. The domain and applications from it should be elaborated.
(2.3) Using GraphQL it is possible to create very flexible APIs and data queries. By implementing it, it should be examined how useful this technique is to limit the transmitted data of a car.
(2.4) A comparison between GraphQL and REST should be created to better understand the pros and cons of both approaches.

REST Representational State Transfer
C&M Cooperation & Management
API Application Programming Interface
IAM Identity and Access Management

[CM-W-CON] Cooperation & Management: CONNECTED CAR, WASA Course Unit. https://team.kit.edu/sites/cm-tm/Mitglieder/2-1.WASA
(1) In order to bring the concept of microservices as small, independent and deployable units to the presentation layer, each microservice should provide its own reusable web-based user interface components. Such UI components should be simple in design and functionality. In addition, UI components from different microservices might use different technologies and frameworks. The challenge is to provide a rich user interface with a good user experience while still maintain flexibility and adaptability for easy modification.

1.1 Different UI components from different microservices must be coupled on the presentation layer in order to provide a rich user interface. Therefore, the presentation architecture contains a message bus to connect UI components in a loosely coupled way.

1.2 A common pattern to decouple components from each other is the publish-subscribe pattern. This pattern allows to add, change or remove components in a fast and easy way.

(2) For a practical understanding on how to build user interfaces for microservices the following work packages are defined.

2.1 A basic introduction on different design patterns for user interfaces, like Model-View-ViewModel and communication patterns, like Publish-Subscribe is required. The most important patterns should be explained and compared to each other.

2.2 A use case for a simple web-based application using the microservice architecture should be designed by following the C&M development process presented in the WASA lecture.

2.2 Web Components [W3C-WebComp] are combining different standardized web technologies to a component model for web applications. No external frameworks are required, therefore development uses plain JavaScript (Vanilla-JS). Some UI components with Web Components should be developed in this work package. As an extension more UI components can be developed with common frameworks, such as Vue.js or Angular.

2.3 In the final stage, the developed UI components should be connected together in a message-oriented way. For this task a simple message bus must be developed in JavaScript for frontend use, which basically allows to publish and subscribe to events.

UI User Interface

(1) The core focus is to extend the already existing C&M development process by a test concept based on a CI/CD pipeline. A CI/CD pipeline is an automated way of doing continuous integration and continuous deployment. Continuous integration means that every committed change will be build and tested directly. Continuous deployment treats every check-in as a release candidate.

(1.1) The key focus of the test concept are E2E tests, which test the functionality of features. Besides that, there are also unit tests, component tests and integration tests. The test concept brings all of these together.

(1.2) Step two is to implement a CI/CD concept for extending the C&M development process.

(2) Work packages for JuniorStudents
(2.1) You will get explained the scope of this project in detail.
(2.3) There are different types of testing for different levels. The goal of this step is to learn them by implementing tests for the project.
(2.4) The BDD features will be tested with another approach. They are mapped to actual feature specifications written in the Gherkin language. To test them, you will learn to automatize different browser by using the Selenium API.
(2.5) The last step is to put everything together and automatize it. You will implement a CI/CD pipeline on GitLab.

CI/CD  Continuous integration/Continuous delivery
C&M  Corporation & Management research group
Cucumber  A tool for testing in Gherkin defined features
Gherkin  A programming language for describing features
GitLab  A web application for Git which offers CI/CD features
Selenium  A tool for testing and controlling web browser

[Sg20] Samuel Gassauer, Development of a Systematic Test Concept as a Fundamental Part of a CI/CD Pipeline, Bachelor thesis, Karlsruhe Institute of Technology, C&M (Prof. Abeck), 2020

Microservice icon: Icon made by Freepik from www.flaticon.com
Quality metrics icon: Icon made by Freepik from www.flaticon.com
Test concept icon: Icon made by dDara from www.flaticon.com
CI/CD Pipeline icon: Icon made by Becris from www.flaticon.com
The core goal is to protect the CI/CD pipeline against attacks and manipulation. A CI/CD pipeline helps to automate the different steps of a software delivery process to bring short and high-frequent development cycles into production after a commit has been issued. The benefit of a pipeline is not only getting earlier feedback after a code change, but also to trace back errors to the developer, who is responsible for the code [HF10].

(1.1) A CI/CD pipeline consists of various components, for example, test and deployment environment, which can have different vulnerabilities. One of the many goals of our project team is to identify the vulnerabilities and to develop countermeasures [Ko20].

(1.2) The developed security concepts should then be applied and evaluated.

(2) The following points show the work packages of the security project team.

(2.1) One of the first goals should be getting a better understanding of the term DevOps and which components play a major role in it.

(2.2) The second step is aiming towards an improved understanding of the terms security and trust and what conditions must be met to establish security with regards to CI/CD pipelines.

(2.3) Threat modeling is an approach to detect vulnerabilities, risks and threats in an application. Threat modeling considers possible entry points from an attacker's point of view. The aim here is to gain basics in threat modeling.

(2.4) From the attacker's point of view, the different security requirements must be identified.

(2.5) The next step is to find and implement solutions to the identified vulnerabilities and risks.

(2.6) Different mechanisms will be developed to warn of and detect potential attacks.

(Illustration) The illustration shows the topics covered by Matthäus Kostka’s Master Thesis.

CI/CD Continuous Integration/Continuous Delivery


The aim of the project team is to improve the qualitative properties of an application. In addition to the improvement of security and scaling aspects, the commissioning and maintenance of the individual services should also be improved. The initial scenario is an application based on microservices.

1.1 The increase of the qualitative properties shall be achieved by the use of service meshes [Cl-Wh].

1.2 The integration of service meshes requires additional services, which act as proxy. Another aim should be the integration of the proxy services into the CI/CD pipeline.

1.3 A further question, which the JuniorStudents should clarify within the framework of their work, is how service meshes can be integrated into existing older applications.

2.1 The JuniorStudents learn the basics of Kubernetes [Cl-Pr] and service meshes [Cl-Wh].

2.2 For the integration and deployment of the applications the JuniorStudents get an introduction to the existing CI/CD pipeline setup at C&M. The setup consists of a Gitlab [Gi-Gi] based build pipeline, which rolls out complete applications into a Kubernetes cluster via a Helm package management [Cl-Th].

2.3 Subsequently, the JuniorStudents should develop a simple application as a service mesh.

2.4 The delivered application is then to be monitored via the control plane. Results should be presentable via Prometheus [Cl-Fr] and Grafana. The gathered information can then be used to scale the single microservices or notify the DevOps team in case of an error.

[Cl-Fr] Cloud Native: From metrics to insight, https://prometheus.io/
The goal of the project is to develop an infrastructure to simulate user interaction (user-generated load) in scalable applications which are still in prototype phase. This allows testing the scalability of the application.

1.1 This infrastructure can be realized using automated load tests, which are scheduled to run at different times per day.

1.2 The tests should be scheduled in a way that the user interactions are simulated as realistic as possible. This allows the machine learning module to deliver realistic predictions.

2.1 Junior Students should get familiar with the concept of performance testing [Wik-Spt], like load and stress testing which is important to accomplish the project task. Additional fundamentals in the Spring Framework [Piv-Spr] are required for the package (2.2).

2.2 The new knowledge about performance testing is to be applied as an example to the Electric Car Charger (ECC) application. It is therefore also important to familiarize oneself with the application

2.3 The student has to familiarize himself with two or more available performance test tools, e.g. Fortio [For-For], JMeter [Apa-Jme], Gatling [Gat-Gat] and present a brief comparison between the considered tools.

2.4 Depending on the results of package (2.3) students have to implement (individually or in a group) a performance test bot for the ECC application to simulate the behavior of users, who are missing in our prototype scenario.

ECC Electric Car Charger

(1) The goal of this project team is to implement monitoring for the TLM application [CM-W-TLM] introduced in the lecture.

(1.1) In traditional, monolith applications, crashes were noticed quickly because all features of the application were affected by a problem. Modern applications, however, are split into several loosely coupled components called microservices. If one microservice crashes, it does not necessarily affect other microservices. Hence a problem might remain unnoticed for quite some time. Therefore, microservices must be monitored (i.e. checked for problem indicators) all the time.

(1.2) Metrics are indicators for the state of a service or its infrastructure. [Dig-AnI]

(2.1) First, you will familiarize yourself with the topic of monitoring. You will find out what monitoring is, what it is used for and which role “metrics” play in this context.

(2.2) Prometheus [Pro-Pro] is a popular tool for monitoring. You will learn how it works, how it can be used, and what kind of monitoring (and different metric types) it supports.

(2.3) You will identify at least three metrics for monitoring the TLM application.

(2.4) You will implement the metrics for the TLM application that you identified before (using Java).

(2.5) The metric implemented in the previous work package should now be monitored by Prometheus. To achieve this, Prometheus needs to be configured to consume (pull) that metric from the application.

(Illustration) The illustration shows the topics covered by Kevin Braun’s Master Thesis [Br20]. The thesis’ goal is to use Machine Learning to improve scaling of applications running in a Kubernetes [Kub-Pro] cluster [You-The].

(TLM App) is an application that serves as a demonstrator for being scaled and monitored.

(Prometheus) is used to collect metrics about the application and about infrastructure related things.

(Autoscaler) is responsible for scaling the application based on Prometheus metrics.

(Machine Learning) is the component that will be developed to analyze the existing metrics, identify trends, and use these trends to create new metrics indicating the estimated future load of the application to allow proactive scaling decisions. These newly created metrics will be passed back to Prometheus in a similar way to metrics used for monitoring applications.

The project team’s contribution to the thesis is that they will figure out how making a metric available to Prometheus works and how Prometheus must be configured to consume a metric. The relevant part of the illustration is surrounded by red dashes.

TLM Todo List Management

(1) The goal of this project team is to achieve loose coupled communication between microservices. Loose coupling means the independence of two microservices communicating with each other [Ne15]. This gives the ability to change a microservice without affecting the counterpart.

(1.1) Asynchronous communication is based on the use of events [BP19]. Microservices only communicate with each other indirectly via another medium.

(1.2) Such a medium is a message broker. It allows to publish microservices on special events or to subscribe to them. Examples of such a message broker are RabbitMQ [Piv-Rab] and Kafka [Ap-Kaf].

(2.1) To work on the project team topic it is necessary to know the basics of loose coupling, asynchronous communication and message broker. To work on the project team topic it is necessary to know the basics of loose coupling, asynchronous communication and Message Broker. Furthermore the event-based communication has to be considered. Within the events there are different concepts for the realization of communication such as event sourcing [Mic-Eve] and saga pattern [Mic-Sag].

(2.2) The new knowledge about microservices is to be applied as an example to the Electric Car Charger (ECC) application. It is therefore also important to familiarize oneself with the application.

(2.3) The message broker has been chosen on RabbitMQ. It is suitable for an easy entry. An instance of the RabbitMQ must be set up for the project team.

(2.4) In order for the ECC to be able to be sensibly based on the RabbitMQ, the application must first be extended by events. These events have to be determined by the project team.

(2.5) If both RabbitMQ and ECC are ready, both can be connected.

ECC

Electric Car Charger

The figure illustrates which research areas are covered by the Bachelor/Master/PhD theses currently worked out at C&M. Each student who takes part in the WASA lecture will be assigned as a JuniorStudent to one of these theses and the related SeniorStudent/PhDResearcher. The SeniorStudent/PhDResearcher is the co-supervisor of the practical/seminar work carried out by the JuniorStudent. A SeniorStudent/PhDResearcher and the JuniorStudents assigned to him/her make up a project team (PT). The PT meetings take place in the Team Room 032 (alternatively in the NZ1 Seminar Room in the case that Room 032 is occupied).

In the following, each PT is presented by introducing into the topic and by showing how the JuniorStudents will be integrated into the work of the SeniorStudent/PhDResearcher who is leading the PT (Schneider, Holzmann, Gassauer, Kostka; Hippchen; Sidler; Hoyer; Throner, Braun, Qattan)

PT  Project Team
This page summarizes all relevant dates that are relevant for each participant of the practical / seminar course offered in combination with the WASA lecture.

(1) It is absolutely necessary that a student has a free slot in his/her personal time table in order to be able to take part in these weekly meetings.

(2) This preliminary version is reviewed by the co-supervising PhDResearcher/SeniorStudent.

(3) The content produced for the presentation should conform to the WASA course material.

(4) The delivery date is the last day of the lecture period.
(1) It is absolutely important that a student who participates in the WASA lecture and practical/seminar course has the necessary resources to cope with the workload (lecture: 120 hours, practical course: 150 hours).

(2.1) Check on the page of the KIT Steinbuch Computing Centre if your email is already depseudonymised.

(2.3) The motivation and the experiences should be summarized in at least one or two paragraphs.

(2.4) This means that the first project team in the list is your favorite team.

(3) This email will be sent by the co-supervisor of the members of the project team which usually is a SeniorStudent.

(3.1) The current lecture material is stored on the C&M Teamserver in the following folder: https://team.kit.edu/sites/cm-tm/Mitglieder/2-0.Aktuelles_Semester
This function can be found in the Studierendeportal by clicking on "Meine Benutzerdaten" > "De-/Pseudonymisierung" and accepting "Ich stimme der Sichtbarkeit meiner namensbezogenen Daten zu". The name-related E-Mail-Adress <prenname><surname>@student.kit.edu" exists additionally to the "uxxx@student.kit.edu" email address.
The C&M Teamserver is described in detail in the document C&M-TEAMARBEIT (in German) which is available
(i) on the C&M web site: https://cm.tm.kit.edu/
(ii) on the C&M Teamserver: https://team.kit.edu/sites/cm-tm/Mitglieder/1-1.Teamarbeit