This WASA course unit [CM-W-INT] describes the content and the organization of the lecture and practical course “Web Applications and Service-oriented Architectures” (WASA) provided by the research group Cooperation & Management (C&M, Prof. Abeck). Current concepts of software development and architectures (including Behavior-Driven Development, Domain-Driven Design, Microservices, RESTful Webservices, 12 Factor App, CI/CD Build Pipelines, DevOps, Container-virtualized Infrastructures) as well as related standards and technologies (including HTTP, Java, JavaScript/TypeScript, Angular, Spring, GitLab-CI, Docker, Kubernetes, Prometheus) are introduced which are needed to develop advanced (i.e. microservice-based, IoT aware, cloud-native, 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.

IMPORTANT: In the winter semester 2021/22, the lecture and practical/seminar course are organized as an online event. This format is called WASAOnline.

The WASAOnline kickoff lecture will take place on Wednesday, 20th October 2021 at 10:00 am.

Each student who wants to take part in WASAOnline and in the kickoff lecture should send an email to cm.research@lists.kit.edu to receive the web link to the web session which is made available for his/her personal use. Please use your depseudonymized KIT student email address (see https://my.scc.kit.edu/shib/pseudonymisierung.php for further information). Thank you!

The lecture material is made available in English. During the online lecture, the content is presented and discussed in German. The oral examination is conducted exclusively in German. Each student writes his/her practical/seminar thesis in English. Thesis templates are made available in LaTeX.

C&M Cooperation & Management  
KIT Karlsruhe Institute of Technology  
WASA Web Applications and Service-oriented Architectures

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

(Connected Car, Healthcare, Environmental) In these business domains 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). All microservice-based applications developed by C&M include aspects of Internet of Things (IoT). An important standard on which our implementations are based on is the SensorThings API from the Open Geospatial Consortium.

(API Management, DevOps, Identity and Access Management) API management can be seen as the bridge between the two C&M's research areas since its task is to manage (i.e., configure, monitor, control) the developed APIs. DevOps concerns the continuous integration (CI) and continuous deployment (CD) of microservice-based software systems into a container-virtualized (Docker/Kubernetes-based) cloud infrastructure. Identity and Access Management (IAM) is a highly relevant crosscutting concern appearing in every web application. An advanced approach is IAM-as-a-service (IAMaaS) by which the IAM functionality is provided in a cloud infrastructure.

BDD  Behavior-Driven Development  
CI/CD  Continuous Integration / Continuous Deployment  
DDD  Domain-Driven Design  
IAM  Identity and Access Management  
IAMaaS  IAM-as-a-Service  
IOSB  Fraunhofer Institute of Optronics, System Technologies, and Image Exploitation  
(Ioptronik, Systemtechnik und Bildauswertung)  
IoT  Internet of Things  
DevOps  Development and 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.

BDD  Behavior-Driven Development
DDD  Domain-Driven Design
These are the microservice-based applications which are developed by C&M in cooperation with his partner institutes.

(IAM Service) The Gherkin features include Identity and Access Management (IAM) requirements, such as authentication of the user of the PCM application. The IAM service Auth0 is used to implement these requirements.

(PCM Microservice, The Twelve-Factor App) The application itself is based on the microservice architecture. The development of the PCM microservice should follow the Twelve-Factor App requirements resulting in a cloud-native application.

(Message Broker, RabbitMQ) The communication between the Backend-For-Frontend (BFF) and microservices is based on the exchange of events. The event bus is provided by a message broker tool named RabbitMQ.

(IoT, IoT Service, SensorThings, Web of Things) IoT standards, such as SensorThings API from OGC and Web of Things from W3C, are taken into account in the development of the IoT-based application.

(API Security) The APIs must be protected against unauthorized access. The IAM service provides the authentication and authorization information needed to take the access decision at the API.

(DevOps, Build Pipelines, Kubernetes Cluster, Prometheus Monitoring) The goal is to develop a build pipeline based on GitLab-CI by which the PCM application is continuously integrated (CI) and deployed (CD) on a Kubernetes cluster. The microservice-based applications provide metrics which are monitored tools such as Prometheus.
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.

COURSE UNIT (P. X – P. Y) All course units written in upper case letter are primary examination material. If page ranges are indicated only these pages are treated in the oral exam. No page range means that questions of the complete course unit can be asked in the oral exam.
The acronym WASA stands for "Web Applications and Service-oriented 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) prosemnal 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 and the proseminal/seminar run in parallel with the lecture course. The practical course counts 5 credit points meaning a workload of 150 hours and the prosemial and seminar count 3 credit points meaning a workload of 90 hours.

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

(3) 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) Therefore, different templates are made available for a practical course thesis and a seminar thesis.
Working in a Project Team

(1) Each project team meeting is prepared by an agenda to which all team members contribute.
(2) Weekly project team meetings starting next week.
(3) Project team members communicate in German.

(1.1) Junior students write their practical/seminar thesis in English and use the LaTeX tool Overleaf to format their text.

(1.2) In the first week all project team members become acquainted with the C&M-TEAMARBEIT document.

The project team agenda is a markdown document by which the work of all project team members is coordinated.

(1) The project team leader takes care that all team members contribute to the agenda of each project team meeting.
(1.1) For each project team a subgroup exists in the C&M GitLab. In the README file of the repository "Projektteamtreffen" the agendas of the project team meetings are collaboratively worked out.
(1.2) The project team meets for one hour every week. The day and time are defined by the project team members.
(1.3) That is why the agenda is worked out in German.

(2) There are templates available for the practical thesis and the seminar thesis.

(3) C&M-TEAMARBEIT is a document [CM-CMT] which describes how the members of the research team C&M efficiently work together.

(Screen dump on the right hand side) This is an excerpt of the GitLab document "Projektteamtreffen" which provides an agenda draft of the first two project team meetings.

Each project team makes a specific contribution to the overall microservice-based system environment.

(PT Schneider) PredictiveCarMaintenance, ClinicsAssetManagement, Microservice Engineering, IoT, SensorThingsAPI, Data Protection

(PT Sidler) EnvironmentalMeasurementDashboard, Domain/Application Microservices

(PT Sänger) MicroserviceDeveloperPortal, API Management, IAM

(PT Throner) DevOps, DevOps Templates, Pipelines, Kubernetes Cluster
(1) The goal of subteam1 of PT Schneider is to provide the current position of a medical device to the ClinicsAssetManagement (CAM) application.

(1.1) For this purpose, the position of the medical device must be known. If a device is moved to another location, the current position must be adjusted. Sensors of a medical device provide the geolocation. This geolocation format needs to be mapped to the required data format of CAM. For this purpose, the microservice SensingDevice was developed in the previous semester. In the next step, the application microservice MedicalDeviceLocator requires to be developed. Furthermore, additional IoT functionality which processes IoT data should be researched and developed within this project team. One seminar goal is to research which IoT functionality is suitable as domain microservice.

(1.2) The IoT functionality should cover analytics and logging of the domain IoT. For example, it is interesting how many information is gathered from the sensor in the last hour. The goal of the project team is to develop one of those services.

(2) Subteam2 (Zingg) is concerned with developing the capability "Management of Vehicles".

(2.1) One task contains the development of the capability management of vehicles by following the microservice engineering process. For the implementation, the implementation and test concept (with focus on the tests) is applied. Therefore, application constraints are defined which are used to derive tests. Problems and insights should be stated and suggestions for improvements are part of this task.

(2.2) Especially integration, consumer-driven contract tests, and end-to-end tests should be implemented according to the concepts if the implementation and test concept. It is important, that the tests run in a CI/CD pipeline. Furthermore, cross-triggering of the pipelines is required which is implemented by the project team.

(3) The task of subteam3 is to ensure that the implementation follows the microservice engineering approach, and especially the software quality during the implementation phase.

(3.1) As a result, the structure of the code (source folder) and the code quality need to be automatically ensured. Therefore, SonarQube should be used as a tool to ensure the quality. Next, the tool should run in a GitLab pipeline to ensure that the repository still follows the guidelines after each commit.

(3.2) Next to the structure, code quality should be measured as well. This part should also be realized with SonarQube. For example, the quality should ensure that method names are meaningful and if common check style (e.g., eslin) are properly used during the implementation (and the code). The results are to be recorded and discussed in the project team. What software quality includes is covered in a (pro-)seminar.
The first task is the development of the application microservices. They are based on the frontends of the LUBW environment applications "air", "flowing water" and "radioactivity". This includes the following steps:

(1.1) An analysis of the corresponding frontends is required. In this analysis, all the potential functionality for the microservices has to be identified.

(1.2) This step deals with the implementation of the mentioned functionality from the first task. For this step, it will be necessary to generate test data as well which deliver input for the application microservices. Later, the application microservices will be connected to the improved domain microservices. During this step, experimental frontends can be implemented, too.

(1.3) In the aftermath of the previous implementation, an analysis of the created implementations is done. Similarities between the microservices (and consequently between the different frontends) have to be found as well as metrics which help to measure the similarity. For this step, additional support is given by the project team leader.

(2) The second task deals with the development of the domain microservices and the according artifacts. This includes:

(2.1) The analysis artifacts have to be adapted to meet the changed requirements for the EMD application, which means the additional LUBW applications which have to be considered ("flowing water", "radioactivity"). Especially, the capabilities have to be adapted and other artifacts like UI flows and features have to be adjusted accordingly.

(2.2) The API and the documentation have to be improved and updated. Based on the adapted artifacts, changes have to implemented.

(2.3) This task is referred to the current status of the implementation. It has to be extended by all the features which were specified in the previous tasks.

<table>
<thead>
<tr>
<th>EMD</th>
<th>Environmental Measurement Dashboard</th>
</tr>
</thead>
<tbody>
<tr>
<td>MS</td>
<td>Microservice</td>
</tr>
<tr>
<td>LUBW</td>
<td>Landesanstalt für Umwelt Baden-Württemberg</td>
</tr>
</tbody>
</table>
Identity and Access Management (IAM) is not yet in practical use at C&M. The goal of the subteam 1 (Ricker) is to add authentication and authorization to the MicroserviceDeveloperPortal (MDP). At C&M, Keycloak is used as an IAM system.

(1.1) The authentication and authorization methods must be added to the frontend and backend services of MDP. This includes a login flow using OpenID Connect (OIDC).

(1.2) The goal is to develop a cross-cutting domain IAM which can also be used by other applications. The gained insights of the subteam Ricker in adding IAM functionalities to an application are then used for a first draft of a cross-cutting domain IAM. One seminar goal can be to research IAM functionalities and support the modelling of a cross-cutting domain IAM.

(2) Subteam 2 is concerned with the further development of the API management capabilities of the MDP.

(2.1) The MDP includes a first implementation of the capability "Management of APIs" which is untested and unfinished. Currently, the MDP can only register and unregister an API.

(2.2) The configuration of the API gateway Tyk through the MDP must be implemented.

(2.3) A frontend for the capability "Management of APIs" to manage API versions and an API gateway configuration must be added to the MDP.

(3) Subteam 3 deals with the extension of the CI/CD pipeline for the MDP. The pipeline extension is called RegisterStep and is used to register new microservices and APIs to the MDP.

(3.1) The current version of the RegisterStep requires an access token for a GitLab repository to register a microservice. This must be changed in a future version of the RegisterStep. Furthermore, the placement of the RegisterStep in the MDP and the ServiceEnvironment has to be further investigated.

(3.2) The placement of the RegisterStep comes along with the revision of the modelling of the domain ServiceEnvironment. This includes rethinking the current subdomains and the placement of API management functionalities.

API Application Programming Interface
IAM Identity and Access Management
MDP MicroserviceDeveloperPortal
OIDC OpenID Connect
Cloud services are currently one of the biggest accelerators for digital transformation. Due to the classic separation of development and operations, the provisioning of these is still often carried out separately from development. This can lead to problems in the coordination and configuration of the services. Accordingly, the goal of subteam1 of project team Throner will be the integration of cloud services into the software development process.

1.1 For the provisioning of cloud services, the infrastructure-as-code approach is mostly used. Within the scope of the internship, common tools for this purpose will be investigated and compared.

1.2 In order to integrate cloud services into the development process, a process needs to be defined which allows developers an easy configuration and integration into their services. As a base the template-based DevOps approach from C&M should be used.

1.3 The Fast Healthcare Interoperability Resources (FHIR) service of the MedicalDevice microservice shall be provided as a cloud service.

Internet facing applications provide a large attack surface for hackers. To prevent potential attacks, it is important to regularly update the application and close potential vulnerabilities. The subteam2 (Petrovic) of the project team Throner deals with the automatic detection and elimination of these vulnerabilities.

2.1 In order to detect security vulnerabilities in the system, it is necessary to read the image of the containers and the dependencies installed within the containers and compare them with a Common Vulnerabilities and Exposure (CVE) database. Each CVE describes a vulnerability and its impact on security. If there is a patch for the vulnerability, it will also be listed in the CVE.

2.2 To make developers aware of the vulnerabilities, all vulnerabilities a microservice has should be displayed in the MicroserviceDeveloperPortal.

2.3 An automated process shall be implemented which automatically fixes the vulnerabilities in the running systems. To do this, all possible patches must be extracted from the previously persisted security vulnerabilities and transferred to the source code of the services.

AWS Amazon Web Services
CVE Common Vulnerabilities and Exposure
FHIR Fast Healthcare Interoperability Resources
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. The meetings start in the next week (i.e. the second week of the lecture period).

(2) The precise date is communicated in the project teams. The date is also fixed in the team calendar of the C&M Teamserver.

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

(4) This day is the Friday of the last lecture week. The whole Friday (i.e. 12 pm) is available to finish the documentation.
(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 depseudonymized.
(2.1.2) The motivation and the experiences should be summarized in at least one or two paragraphs.
(2.1.3) 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.
The current lecture material is stored on the C&M Teamservier in the following folder:
https://team.kit.edu/sites/cm-tm/Mitglieder/2-0.Aktuelles_Semester
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
Activation of the Name-Related E-Mail Address (Depseudonymization)

Can be carried out via Shibboleth
(https://my.scc.kit.edu/shib/pseudonisierung.php)

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". Die name-related E-Mail-Adresse <prenom><surname>@student.kit.edu" exists additionally to the "uxxx@student.kit.edu" email address.