ZVAX – A Microservice Reference Architecture for Nation-Scale Pandemic Management

Oliver Cvetkovski
Zurich University of Applied Sciences, Winterthur, Switzerland
University of St. Cyril and Methodius, Skopje, North Macedonia

Carlo Field
Zurich University of Applied Sciences, Winterthur, Switzerland

Davide Trinchi
Zurich University of Applied Sciences, Winterthur, Switzerland

Christof Marti
Zurich University of Applied Sciences, Winterthur, Switzerland

Josef Spillner[1]
Zurich University of Applied Sciences, Winterthur, Switzerland

Abstract

Domain-specific Microservice Reference Architectures (MSRA) have become relevant study objects in software technology. They facilitate the technical evaluation of service designs, compositions patterns and deployment configurations in realistic operational practice. Current knowledge about MSRA is predominantly confined to business domains with modest numbers of users per application. Due to the ongoing massive digital transformation of society, people-related online services in e-government, e-health and similar domains must be designed to be highly scalable at entire nation level at affordable infrastructure cost. With ZVAX, we present such a service in the e-health domain. Specifically, the ZVAX implementation adheres to an MSRA for pandemic-related processes such as vaccination registration and passenger locator form submission, with emphasis on selectable levels of privacy. We argue that ZVAX is valuable as study object for the training of software engineers and for the debate on arbitrary government-to-people services at scale.

2012 ACM Subject Classification Computing methodologies → Distributed computing methodologies; Computer systems organization → Availability; Applied computing → E-government

Keywords and phrases Nation-scale applications, microservices, digital transformation

Digital Object Identifier 10.4230/OASIcs.Microservices.2020-2022.4

Related Version *Previous Version*: `https://www.conf-micro.services/2022/papers/paper_6.pdf`

Supplementary Material *Software*: `https://doi.org/10.5281/zenodo.7869751`

1 Introduction

Microservice-based software applications are expected to have many advantages. They are supposed to be easier to develop with distributed teams of software engineers using polyglot implementations, to allow for more customisation through flexible service composition, and to be more aligned with business-critical runtime properties such as high scalability and resilience.

[1] Corresponding author

Joint Post-proceedings of the Third and Fourth International Conference on Microservices (Microservices 2020/2022).
Editors: Gokila Dorai, Maurizio Gabbrielli, Giulio Manzonetto, Aomar Osmani, Marco Prandini, Gianluigi Zavattaro, and Olaf Zimmerman; Article No. 4; pp. 4:1–4:16
OpenAccess Series in Informatics
OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

In the first years of research on microservices, practical applications to prove architecture-related hypotheses and to implement new and innovative concepts were lacking. A first applications overview was assembled in 2017 [7]. Since then, the knowledge on microservice architectures has increased. This is especially true for microservice reference architectures (MSRAs) that serve as blueprint for other applications in the same domain. Multiple studies have been supported by the growing insight into such architectures, the coupling of the affected services, and the messaging patterns between them, as well as the resulting runtime characteristics. For instance, six reference architectures and use cases were analysed for service dependencies and interchangeability in software product lines, including a detailed study of the demonstration microservices of the Google Cloud Platform [3]. A well-known reference architecture for e-commerce is the Sock Shop, despite not having been formally introduced in the literature, due to its popularity in diverse software modernisation trainings. It consists of six polyglot backend services and one queue and is publicly available on the web[2]. Sock shop has been used among other works by the authors of MicroRCA, a root cause analysis framework to spot performance issues in microservices [30].

More focused and domain-specific understanding of the runtime properties of applications become feasible when further MSRAs are investigated and published. In recent years, this has increasingly been the case. For instance, MSRA for measurement systems and enterprise measurement infrastructures were designed and evaluated [28]. Such systems collect business metrics along with business insights and feed them into enterprise dashboards. Advanced measurement systems include semantic measurement models with distributed data management [5]. The SAMSP platform for self-adaptive microservices has been similarly published. It supports instance overload, unreachable services and other events to adjust service delivery [17]. A multi-tenant SaaS hosted in a cloud based on microservices has been validated by transforming Microsoft MusicStore [24]. A platform based on ProteomicsDB has been built to investigate the use of microservices for proteomics and personalised medicine [23]. Elasticity research in document management systems and the the investigation of service level objective violations in such elasticity scenarios yielded additional MSRAs [18, 25]. Even combined reference architectures for microservice- and blockchain-based applications have been proposed and studied [29].

While the existing MSRAs cover many domains, they are generally limited to business applications with a small or undefined set of users. Moreover, as noted by the authors of one of the elasticity works [25], they are often not built on state-ofthe-art technologies, and lack a description of asynchronous communication patterns and adaptation mechanisms that are essential for achieving scale especially in overload situations. Following the increasing digitalisation and digital transformation in society, nation-scale applications are emerging and thus warrant further analysis concerning their realisation based on microservices. This concerns in particular e-government applications addressing all citizens, residents and visitors to a country within short time periods, demanding massive elastic scalability in compute services but also adequate application design to reflect such computational capabilities.

In this article, our open source ZVAX implementation [10] shall serve as exemplary nation-scale application for the domain of digital pandemic management, and as realisation of a corresponding RA. The work consolidates our preliminary research conducted in previous years [11, 9] and adds more details, in particular a performance analysis related to deployment in local and public container platforms, and a generalisation of the architecture towards concerns found in other e-government applications. We consider ZVAX to benefit from

[2] Sock Shop website: `https://github.com/microservices-demo/microservices-demo`

a reference architecture for such applications due to the careful design and realisation of the underlying microservices. It is a fully functional system covering multiple pandemic management fields with separation of concerns along service boundaries. Hence, we argue that ZVAX serves as practical study object in microservice-related education, such as courses in software architecture or cloud-native application development. In the next sections, we first give a background on the emergence of nation-scale applications across several areas but with emphasis on e-health and pandemic management. Then, we derive the MSRA methodologically, present the microservice concepts of ZVAX and explain the realisation of the underlying MSRA on the implementation level. Moreover, we evaluate its combination of harmonic scalability and selective decentralisation. Finally, we motivate further research directions.

2 Background: Nation-Scale Applications

We define the term nation-scale application to refer to any software application that needs to scale to the population of a country, not necessarily in terms of concurrent users but within upper time boundaries that are often mere minutes. Often, these applications are in the domain of e-government addressing people, especially G2C (government to citizens) or G2P (government to citizens, residents, visitors and the general public). Evidently, there are commercial applications from private operators with even larger, de-facto global reach, but in those there are no critical moments when all users need to be reached within a small bounded time window. There are also larger applications not involving users based on machine communication (M2M, M2G), but due to being non-interactive, moderate delays are less critical and do not end up in emotionally driven overload amplification out of fear or panic. Our definition is the first to apply to software applications, whereas we would like to point out that nation-scale systems engineering has been investigated before concerning waste management, building modelling, land surveys and similar engineering concerns [20, 4], as well as batch data processing [19].

Nation-scale applications require a high-bandwidth, low-latency communication channel for broadcasting and point-to-point messaging. They are particularly linked to mobile devices, mobile telecommunications and cloud infrastructure due to the ability to trigger spike-capable access to the application within a short time period. Cell broadcast mechanisms are available and have been investigated for cloud integration [16] but are confined to non-personalised content distribution such as general information to the population in multimedia formats. Fig. 1 expresses the relationship between infrastructure, devices and applications. In the following, the characteristics of four nation-scale application areas – emergency alerts, connection inquiries, crisis expert search and finally pandemic management – are explained to determine commonalities for any nation-scale MSRA. Empirical evidence is collected from the respective Swiss and European population. The pandemic management application area is furthermore discussed in detail to give sufficient insights into the domain chosen for our specific MSRA.

The first area for applying nation-scale is in public safety, in particular emergency alert applications. A recent comparative study has compared several of those in Europe [15]. The study omitted a technical analysis of how scale is achieved on the software side but gives insight into the scale needed for crisis and hazard communication. It also documents how location-based SMS and cell broadcast is used for crisis communication, given that a false positive (a person receiving a message in error) is more tolerable than a false negative (an intended recipient not being informed in time). AlertSwiss, the emergency alert mobile application for Switzerland and Liechtenstein, has disseminated around 1200 messages over

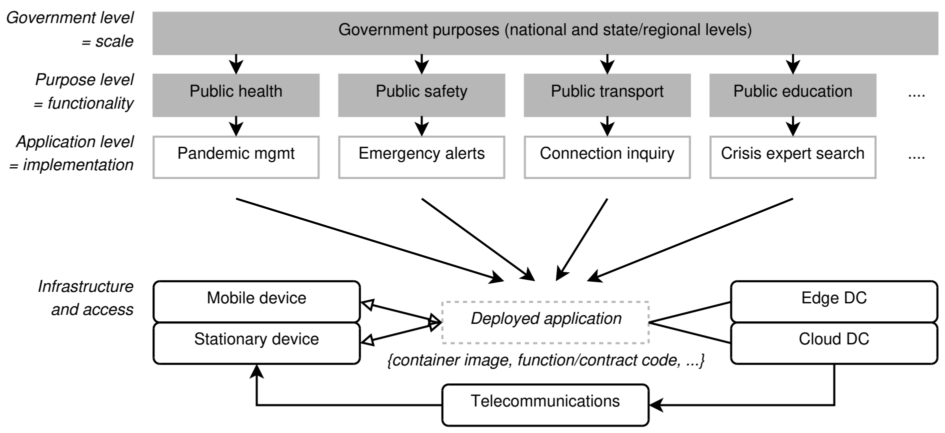


Figure 1 Infrastructure positioning of nation-scale application areas.

five years, and is supposedly reaching around 12% of the population, bounding the nation-scale level at one million people within a timeframe of second to minutes for severe notifications such as earthquakes.

The second area is in public transport, in particular connection inquiry applications. Such applications are among the most-used ones by the population due to the everyday importance of mobility. In countries providing an integrated national transport system, the acceptance and adoption is particularly high by also reaching risk-averse passengers [13]. According to a survey conducted among our students, about 94% occasionally or regularly use the timetable updates provided by the Swiss public transport system. The application is running as containerised workload across two data centres, without public numbers on associated cost or scalability characteristics. The nation-scale level can be assumed to be several million people with however a larger time window, often in the order of hours for trip planning or discounted ticket sales.

The third area is in the public education system, in particular the search of experts on a national level in crisis situations. While today, most education and training certificates are handed out on paper or as human-readable electronic documents, the trend towards microcredentials [21] is demanding new software architectures to handle automated fine-grained creation and verification of those credentials, and complex expert search and skills matching microservices on top of this information base. The Swiss public university system in its foundational programmes alone produces around 60000 certificates covering around 2 million grades which can be estimated to be broken down to more than 10 million microcredentials per year with clear peak service times tied to the semester schedules but also to crisis hiring situations.

The fourth and more deeply investigated area is in public health, in particular pandemic management, which is related to highly sensitive personal data and therefore only few messages, such as general information about pandemic spread, can be considered for broadcasting. During the COVID-19 pandemic, pandemic management services addressing the entire population were provided but often at high cost for raw data centre resources such as virtual machines, without reconsidered and rethought software architecture. In other instances, scalability problems such as downtimes and long delays occurred and made it into the national press, contributing to the population's anger about the political management of the pandemic[3]. Modern public cloud service models are promising high scalability and appear

[3] Swiss Radio & Television, Espresso, January 15, 2021

to be a solution. They are nevertheless only a part of a solution. First, public cloud services are often also limited to few thousand concurrent instances, which is insufficient for the problem field where tens of thousands of parallel requests may arrive. Second, they provide the heavy lifting but require domain-specific glue logic and interaction patterns, an area where much of the mistakes in architecture design may occur. The blueprint knowledge on how to build nation-scale microservice applications in the e-health and pandemic management area is thus limited to few approaches on vaccination passports [2, 12, 8].

3 Related Work: Reference Architectures and Benchmarks

Beyond the application area of pandemic management, software engineering research specifically aiming at the composition of applications from microservices has produced a wide range of reference architectures, both abstract ones and others applicable to particular domains. As previously mentioned, this encompasses for instance building modelling, land surveys and batch data processing [20, 4, 19] among others. In order to validate these architectures, in many cases generic experiments and benchmarks have been used by the authors, while in parallel, the research community has also contributed microservice-specific benchmarks.

Among those benchmarks are μ-Suite, Acme Air, DeathStarBench and HydraGen. μ-Suite investigated low latency applications such as image similarity or recommender systems [26] and exposed the need for latency-aware scheduling on the OS level to avoid high tail latencies. Acme Air was originally a monolithic web performance benchmark. It was adapted to Node.js and Java microservice execution [27] with credible reports about the performance dropping to around 20% of the monolithic version due to overheads. DeathStarBench incorporates sample applications such as social networks, media and e-commerce sites, online banking and vehicle control [14]. It is available as open source framework and again reports tail latencies at scale among other metrics. Its authors shifted the emphasis from a comparison against monolithic architectures towards a reasoning about certain performance quirks. HydraGen takes benchmarks a step further and generates them [22]. HydraGen has been validated in traffic engineering but its design does not preclude other application domains.

Hence, it becomes apparent upfront that a pure performance-oriented design might not be best served by microservices. However, elastic scalability beyond machine boundaries and the increasing investments of cloud providers into microservice hosting platforms suggest that the drawbacks might become less significant especially for nation-scale requirements. Consequently, researchers have investigated scaling and scalability properties such as by-design global scaling [1], but also orthogonal concepts such as API patterns including scalability-related quality patterns [31] and deployment [6].

4 Concepts: Scalable Pandemic Management

We condense the problems raised in the introduction into a single research problem: Which software design and architecture adequately fits the e-health domain and more specifically the task of nation-scale pandemic management? In order to address the problem, we follow a methodology consisting of four steps. In the first methodological step, we contribute four novel concepts as generic, domain-independent architectural design foundation.

1. Nation-scale services. They are built to serve large parts of a population in a short amount of time. In quantitative terms, we assume a lower bound of many 10,000s of short-lived requests per second. This purposefully exceeds current public cloud offerings by an order of magnitude even when combining regions. We note that the target concurrency

can often be influenced by political means outside the technical scope, for instance by population segmentation by age group. Such segmentation may however not be applicable in emergency situations. Under no circumstances should users perceive downtime or unresponded hanging requests, due to the danger of exacerbating the system load by follow-up actions in panic.
2. Harmonic scalability. A distributed system is harmonically scalable if all of its constituent parts scale along the critical request paths. For an architecture based on microservices, this entails the ability to scale elastically in each service, but also in attached middleware services. In geometric terms, the system representation may overall shrink or grow but the proportions remain the same.
3. Selective decentralisation. The scalability is influenced by user preferences on where to store and process data. Hence, decentralisation is used to reduce microservice invocation load at neuralgic points while at the same time offering stronger privacy guarantees on demand. From an architectural perspective, the selective decentralisation leads to a selectively externalised statefulness, referring to the state as output of one microservice that determines the follow-up behaviour of another microservice.
4. “Flatten the curve”. This term originating in the domain-specific goals of pandemic management also applies to the prevention of microservice overload. Queues and other asynchrony mechanisms are used to facilitate quick responses to service requests, leaving the bulk of the work for less loaded periods of time while granting a rapid responsive behaviour to users navigating the user interface. Users are then informed asynchronously at later points in time about the results of compute-centric tasks through notification channels depending on the registered contact details.

Next, we derive a domain-specific reference architecture for the domain of pandemic management. The architecture must support the functional requirements implied by the domain, and adapt to the underlying infrastructure especially in terms of computation and communication. A representative application covering the main governmental interests in pandemic management consists of the following four classes of services to the population.

1. Appointments. In order to prevent people from queueing unnecessarily, appointments help to “flatten the people curve” in real-world situations such as testing and vaccination points.
2. Certificate creation and signing. Various schemes exist, with most having settled on a QR code representation digitally signed by a single authority or, for better fraud protection, multiple authorities.
3. Certificate checking. The inverse process, combining signature validity checks with expiration policies.
4. Information solicitation. This encompasses mandatory solicitations such as passenger locator forms before arriving in a country (triggering a certificate creation) and contact quarantine tracing upon cases of assumed or confirmed infections, but also voluntary information provided to assist the tracing.

These four classes of services need to be mapped to four or more technological realisations. In a third methodological step, we therefore combine the conceptual requirements and functional scope, and match them against recent technological progress. This means that all the management services of the four classes are instantiated and delivered with high elastic scalability and low latency in order to achieve the desired volume of requests. Four main technological advances are increasingly available in managed microservice environments and are considered favourable from the reference architecture perspective.

1. Multi-core function runtimes and container-native hosting for maximised local parallelism, attached to low-latency local storage such as RAM and NVRAM. Often, the underlying execution follows a uniform model based containerisation or hardware-accelerated virtualisation such as ARM TrustZone/vTZ, AMD-V or Intel VT-x. To the software engineer, the execution offers deployment interfaces as raw container images (following the de-facto standard set by the Open Container Initiative), function source code (FaaS with its many language- and provider-specific syntax requirements and constraints), source code in microservice-oriented languages (e.g. Jolie), or smart contracts in managed blockchains (e.g. canisters). All non-image software artefacts are automatically converted to appropriate images upon deployment, keeping the engineer free from infrastructural concerns but also limiting potentially performance- or latency-improving tuning. We consider all of those interfaces valid technological choices for a microservice design as long as service-oriented characteristics (well-defined interface supporting one or more message exchange pattern through loose coupling) are fulfilled. Containers, in particular, do not have an inherent service orientation but are suitable to encapsulate one or multiple services.
2. Event-driven asynchronous function execution. Microservice instances, in particular when designed as event-triggered functions, scale almost proportional to the request rate. Factors such as cold start (for initial invocation) or spawn start (for concurrent invocation) have been thoroughly investigated in recent years and are now well understood, and with methods such as prewarming and idle time optimisation, further scalability gains can be achieved.
3. Preparedness for extreme edge deployments. New hardware allows for deploying microservices directly on network interfaces (e.g. SmartNICs) in edge data centres, allowing for unprecedented scaling of stateless services by geographic distribution close to the points of use and their low-latency delivery. This can be exploited for instance to distribute the work-intensive QR code creation and signing processes that require no state other than a set of input parameters. Moreover, federated deployments become possible to map political hierarchies and pandemic management responsibilities (e.g. federal-level, state-level) to delivery locations. At the same time, the system needs to remain deployable on a single off-the-shelf device to achieve full elasticity from single developer to nation-scale.
4. Flexible bindings to communication infrastructure. For instance, a service can bind to telecommunication cloud services to obtain a regional cell broadcasting interface, or to satellite providers for full global coverage but with narrowband links. Alternatively, it can use conventional mobile backend-as-a-service (MBaaS) interfaces for personalised push notifications. In conjunction with edge deployments, these topological concerns are of interest when considering nation-scale beyond the mainland boundaries of countries, in particular for island nations or when governments want to address their citizens abroad. In a Swiss context, around 9% of the population or 800000 citizens live abroad and might need to be included in short-lived e-government processes such as electronic voting, but also in long-term services that benefit from follow-the-sun microservice provisioning semantics across federated clouds or edges.

The concrete management system reference architecture results as a fourth step of the methodology. It is synthesised along with constituent microservices in Fig. 2, starting from the user perspective of either personal access through a mobile device or computer, or a stationary device such as a QR code scanner located at the entrance of a location that mandates such checks – such as restaurants, public authorities or university campuses.

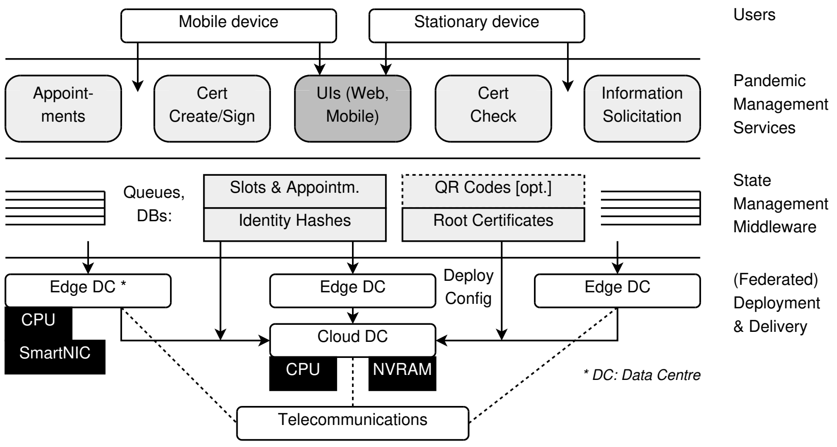


Figure 2 Reference architecture; components marked [opt.] are optional and only activated on demand depending on user preferences due to being offloadable to client-side devices.

The architecture reflects the ability to choose, on a per-user basis, whether to opt into more government-managed or more self-managed data records, with corresponding responsibilities for backups and access protection. Moreover, it aligns with the increasing availability of micro datacentres and other edge deployments for scalable service delivery close to the users, as well as different capabilities in telecommunications infrastructure, including selective availability of cell broadcast and often still limited access to such facilities from cloud APIs.

5 Implementation: ZVAX

ZVAX, the Zurich vaccination and pandemic management software [10], is a fully functional demonstration system of the MSRA for nation-scale pandemic management. Its development started in the second year of the COVID-19 pandemic, and its functional extensions were driven by digital management processes introduced by governments around the world, with emphasis on the services on different administrative levels of the Swiss confederation, Northern Macadonia, and other European countries. Hence, apart from implementing the MSRA and thus overcoming scalability and privacy issues, it also serves as testbed for solving challenges that emerged on a societal level in Switzerland, often with extensive local or national press coverage. These challenges relate to many cases of certificate fraud (solved in ZVAX by multi-signatures, leading however to larger QR codes), interoperability (solved by multi-QR code schemes), different expectations on privacy (solved by selective decentralisation), and flexible federated deployment (solved by an adaptive user interface connected to configurable sets of backend services). ZVAX can furthermore serve as testbed for improved client-side QR code detection, for instance on black background that usually causes problems in today's mobile applications, and coloured or animated codes to represent portrait photos for safer identification.

The microservice architecture of ZVAX is shown in Fig. 3. The figure omits the information solicitation service to focus on those of relevance for the most complex workflow: making an appointment for vaccination, creating and signing a certificate during vaccination, and checking the certificate afterwards.

Fig. 4 gives an impression of the user-facing functionality of the current implementation. Users are able to select the desired level of privacy, in turn determining the degree of centralised versus decentralised data storage. Decentralisation is achieved by a combination

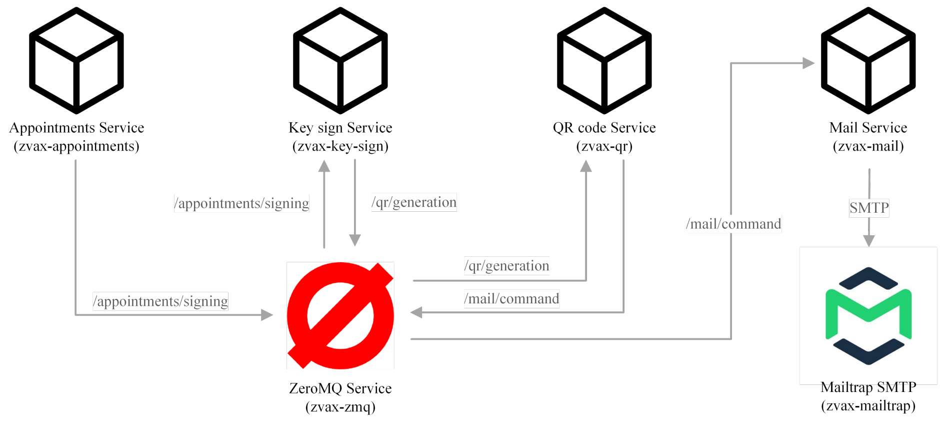
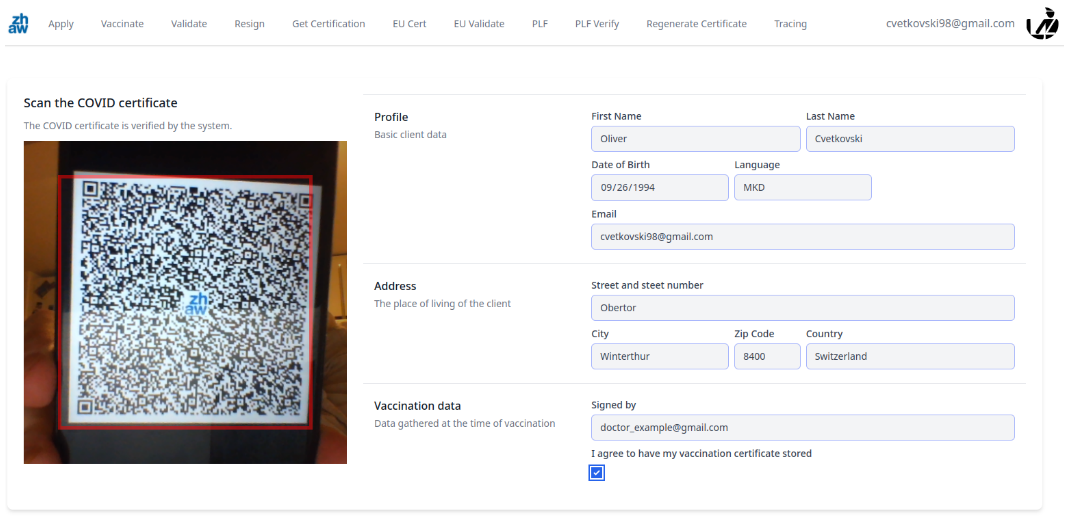


Figure 3 Microservice composition of the ZVAX implementation.

of in-browser storage and file downloads, leading to no trace of any health-related activities when full decentralisation is selected by the user. Subsequently, users apply for a test or vaccination appointment, retrieve certificates, enter locator form information, or manage contact tracing. Correpondingly, doctors and health officials are able to sign (and counter-sign) as well as verify certificates.



Scan the COVID certificate

The COVID certificate is verified by the system.

Profile

Basic client data

First Name: Oliver

Last Name: Cvetkovski

Date of Birth: 09/26/1994

Language: MKD

Email: cvetkovski98@gmail.com

Address

The place of living of the client

Street and steet number: Obertor

City: Winterthur

Zip Code: 8400

Country: Switzerland

Vaccination data

Data gathered at the time of vaccination

Signed by: doctor_example@gmail.com

I agree to have my vaccination certificate stored [✓]

Figure 4 Exemplary screenshot of the QR code verification within the ZAX implementation.

Appointments are possible for groups of persons, reducing the need to fill out forms considerably for families. The appointments-related invocation flow on the microservice level encompassing the above-introduced microservices, not including the middleware components, is shown in Fig. 5. All microservices can be scaled with instance selection through the Traefic load balancer. First, the authentication service grants a time-limited token. Then, the appointments service is invoked with the token. In case an appointment can not be obtained immediately or the system is overloaded, an asynchronous re-invocation is scheduled ("flatten the system load curve"). Next, the appointment is expressed as a signed QR code, re-using the same services that will also perform the same task for vaccination certificates.

All personal information apart from the time slot blocking and an identity hash are then removed from the system, and the QR code is sent via e-mail for the person(s) having received an appointment.

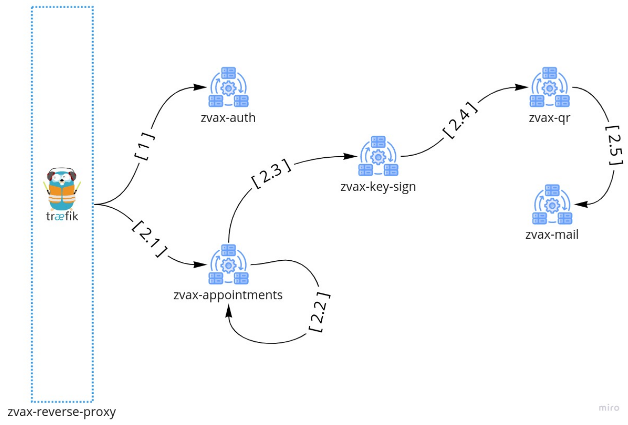


Figure 5 Exemplary microservice invocation flow consisting of two subflows and six steps.

The ZVAX implementation is easy to bootstrap on new machines due to its complete containerisation. The codebase is prepared to be scaled horizontally with orchestrators such as Docker Compose, Docker Swarm and Kubernetes. Almost all microservices ship with a single implementation. In order to stress the substitution principle and to compare polyglot implementations, the QR code microservice ships with two implementations, in Python and in Rust. This has not only performance implications, but also affects the scaling behaviour due to differences in size and resource requirements of the resulting container images. We have carefully designed both the individual microservices and the entire composition to adhere to harmonic scaling. Often, this term refers to harmonic local-versus-global scaling in the general sense [1] whereas we specifically followed the request path and ensured that no bottleneck would occur that could starve subsequent processing steps along the same path.

6 Evaluation: Performance and Scalability

We have evaluated ZVAX to demonstrate its preparedness for nation-scale deployments, primarily by taking performance and correctness measurements at different active user scale levels. To compare modern cloud-native microservice deployments including Function-as-a-Service (FaaS) and auto-scaled Kubernetes instances, we have set up a scalable testbed on large virtual machine instances on the OpenStack cluster 'apu' in Zurich University of Applied Sciences. The testbed consisted of a single master node and three worker nodes, from which the results of larger deployments can be interpolated. ZVAX got deployed to the infrastructure via Helm. In addition to the system under test, the cluster hosts a set of auxiliary services to better diagnose runtime results. This includes a monitoring stack using Grafana and Prometheus, an alternate autoscaler called KEDA (Kubernetes Event-Driven Autoscaling), OpenFAAS and Cert-Manager. Grafana and Prometheus provide dashboards that display metrics about the cluster, K6 tests, Keda and RabbitMQ. Cert-Manager is used to provision certificates (related to the infrastructure communication, not related to pandemic management) automatically. The entire setup is shown in Fig. 6.

Spike and stress testing experiments are done by using K6 by Grafana. The tool allows us to design detailed scenarios that are executed onto the chosen service endpoints via simple HTTP requests. Prometheus is used as a monitoring solution to analyse how the services

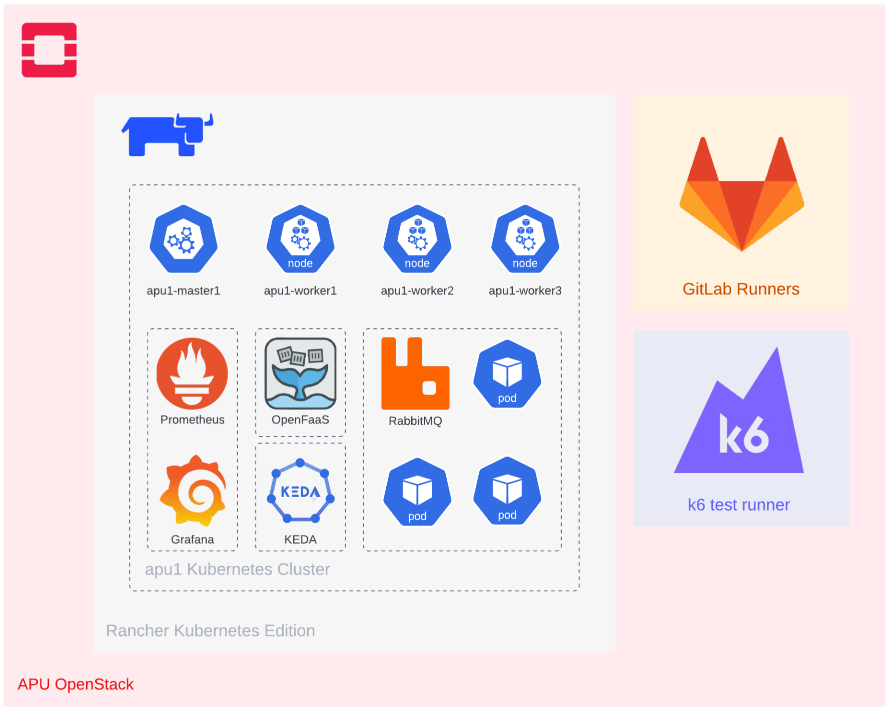
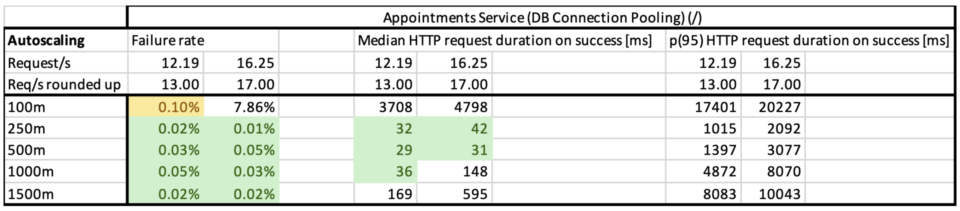


Figure 6 Testing and evaluation infrastructure.

behave regarding CPU and memory consumption. Tests are per service and endpoint. Each load test is tuned to that specific service, as not all services have the same load and demands in a real-world scenario. Estimated loads are determined using available COVID-19 datasets and both measured and estimated computational complexity. Tests are run using variable scaling. Service replica count and CPU limits can be tuned for each test using Helm. The evaluation aims to answer three main questions: (i) How many resources does an individual service instance require?; (ii) How do we combine variables to achieve optimal scalability?; (iii) Where are bottlenecks? Load profiles are established using estimated load, ranging from 0.25x (tier 1) to 2x (tier 6). The experiments encompass the time required for QR code generation, certificate signing, certificate storage, appointment booking, as well as alternative QR code generation through OpenFAAS, the Rust-based implementation and KEDA, and alternative appointment booking through connection pooling.

Due to limited space, not all experiments can be replicated here. We include two interesting results that show the benefits of the chosen concept. First, the appointment booking with connection pool of size 20 with a buffer of 40 (Fig. 7) offsets a high error rate that is otherwise introduced by autoscaling, by harmonising request queueing and processing. The mean failure rate drops from around 4% to around 0.10%.

Second, the QR code was generated following the Swiss distribution of PCR tests with a peek rate of 2.48 per second and vaccination/booster with a peek rate of 2.08 per second, totalling 4.56 requests/s, and when assuming an eight-hour work window for vaccination and test centres, 6.84 requests/s (RPS). Fig. 8 compares the median HTTP request duration for (a) static replica counts and (b) horizontal autoscaling (HPA). On tier 3 as comparison point, autoscaling achieved 7 RPS, sufficient for the peek rates, with a replica CPU limit



	Appointments Service (DB Connection Pooling) (/)								
Autoscaling	Failure rate			Median HTTP request duration on success [ms]			p(95) HTTP request duration on success [ms]		
Request/s	12.19	16.25		12.19	16.25		12.19	16.25	
Req/s rounded up	13.00	17.00		13.00	17.00		13.00	17.00	
100m	0.10%	7.86%		3708	4798		17401	20227	
250m	0.02%	0.01%		32	42		1015	2092	
500m	0.03%	0.05%		29	31		1397	3077	
1000m	0.05%	0.03%		36	148		4872	8070	
1500m	0.02%	0.02%		169	595		8083	10043	

Figure 7 Autoscaled and connection-pooled appointment booking.

of 1000 millicores, preventing the system from throttling. HPA is a reactive autoscaler based on deferred metrics (metrics server interval + HPA interval + scale-up reaction time), leading to insufficient reaction agility in contrast to anticipating autoscalers. With KEDA, the metrics resolution issue disappears but the remaining slowness, in particular the 15 seconds HPA interval, remains. Moreover, as the chosen scaling setting for KEDA is based on the queue length, sudden bursts of traffic may cause the queue length to climb drastically, as instances cannot keep up with queue intake. In such cases, KEDA may overprovision instances until the system stabilises. As a conclusion from this experiment, we can consider it a positive coincidence that HPA provided sufficient upscaling speeds for the observed peek rate development, but for safe-guarding similar scenarios in which anticipation is not possible, more rapidly reacting autoscalers and pod schedulers should be developed for Kubernetes.

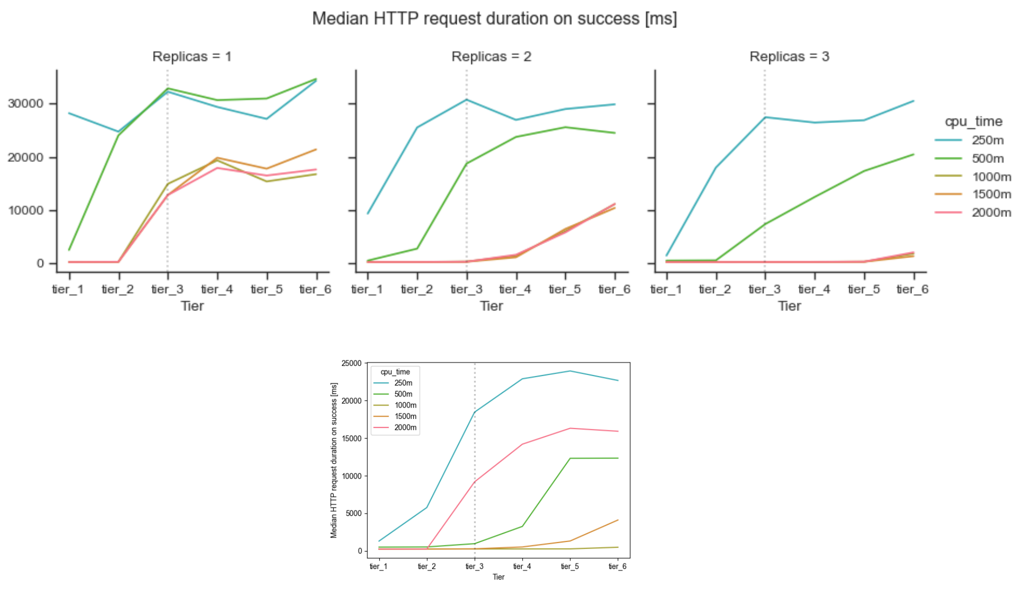


Figure 8 Static and autoscaled QR code generation.

Overall, our architecture has shown to be sufficiently scalable for a country the size of Switzerland on 4 VMs, but required tuning and exploring the different deployment options. The influence of deployment is therefore considered as important as the influence of the microservice design.

A crucial and open discussion point is the generalised design of such software architectures for deployments in arbitrary countries, including those with approximately 100 times the population size. Apart from the mentioned geographic topologies (edge computing, federated cloud or multi-cloud), such designs will likely benefit from hierarchical structures following the administration levels, such as states and union territories in India or provinces, autonomous regions and direct-administered municipalities in China. Fiduciary agreements can then be used to balance operational effort and scalability needs, for instance, by a single cloud deployment covering several smaller states or provinces.

7 Conclusions and Future Work

With the open source system ZVAX[10], we have developed a pandemic management system that improves upon current government-provided services. It is designed to accomodate surge requests and sustained requests at a higher scale, and adjusts to user preferences concerning data handling through selective decentralisation. ZVAX builds upon a domain-specific MSRA that takes current technological progress into account and lets researchers explore more flexible externalised state management beyond the conventional stateful/stateless distinction. Moreover, ZVAX is positioned as study object for the training of software engineers who learn the construction of societally relevant digital services.

Future work is divided into domain-specific and general architectural directions. Within the domain of pandemic management systems, we aim at reducing the need for complementing the QR code with a passport by embedding photographic identity information within a static (coloured) or animated code. On the architectural level, we aim at conducting large-scale and nation-scale performance tests to prove the proposed approach at all levels: services, queues, composition and deployment. This will encompass the acceleration gained by edge deployments, and thus contribute to the debate on digital souvereignty of administrative levels achievable through a wider digital transformation enabled by flexibly deployable microservices.

References

1 Lorenzo Bacchiani, Mario Bravetti, Maurizio Gabbrielli, Saverio Giallorenzo, Gianluigi Zavattaro, and Stefano Pio Zingaro. Proactive-reactive global scaling, with analytics. In Javier Troya, Brahim Medjahed, Mario Piattini, Lina Yao, Pablo Fernández, and Antonio Ruiz-Cortés, editors, *Service-Oriented Computing - 20th International Conference, ICSOC 2022, Seville, Spain, November 29 - December 2, 2022, Proceedings*, volume 13740 of *Lecture Notes in Computer Science*, pages 237–254. Springer, 2022. doi:10.1007/978-3-031-20984-0_16.

2 Masoud Barati, William J. Buchanan, Owen Lo, and Omer F. Rana. A privacy-preserving distributed platform for COVID-19 vaccine passports. In Luiz F. Bittencourt and Alan Sill, editors, *UCC '21: 2021 IEEE/ACM 14th International Conference on Utility and Cloud Computing, Leicester, United Kingdom, December 6 - 9, 2021 - Companion Volume*, pages 16:1–16:6. ACM, 2021. doi:10.1145/3492323.3495626.

3 Benjamin Benni, Sébastien Mosser, Jean-Philippe Caissy, and Yann-Gaël Guéhéneuc. Can microservice-based online-retailers be used as an spl?: a study of six reference architectures. In Roberto Erick Lopez-Herrejon, editor, *SPLC '20: 24th ACM International Systems and Software Product Line Conference, Montreal, Quebec, Canada, October 19-23, 2020, Volume A*, pages 24:1–24:6. ACM, 2020. doi:10.1145/3382025.3414979.

4 Andy S. Berres, Brett C. Bass, Mark B. Adams, Eric Garrison, and Joshua R. New. A data-driven approach to nation-scale building energy modeling. In Yixin Chen, Heiko Ludwig, Yicheng Tu, Usama M. Fayyad, Xingquan Zhu, Xiaohua Hu, Suren Byna, Xiong Liu, Jianping Zhang, Shirui Pan, Vagelis Papalexakis, Jianwu Wang, Alfredo Cuzzocrea, and Carlos Ordonez, editors, *2021 IEEE International Conference on Big Data (Big Data), Orlando, FL, USA, December 15-18, 2021*, pages 1558–1565. IEEE, 2021. doi:10.1109/BIGDATA52589.2021.9671786.

5 Eric Braun. *Microservice-based Reference Architecture for Semantics-aware Measurement Systems*. PhD thesis, Karlsruhe Institute of Technology, Germany, 2020. URL: `https://nbn-resolving.org/urn:nbn:de:101:1-2020112503582180757762`.

6 Mario Bravetti, Saverio Giallorenzo, Jacopo Mauro, Iacopo Talevi, and Gianluigi Zavattaro. Optimal and automated deployment for microservices. In Reiner Hähnle and Wil M. P. van der Aalst, editors, *Fundamental Approaches to Software Engineering - 22nd International Conference, FASE 2019, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2019, Prague, Czech Republic, April 6-11, 2019, Proceedings*, volume 11424 of *Lecture Notes in Computer Science*, pages 351–368. Springer, 2019. `doi:10.1007/978-3-030-16722-6_21`.

7 Antonio Brogi, Andrea Canciani, Davide Neri, Luca Rinaldi, and Jacopo Soldani. Towards a reference dataset of microservice-based applications. In Antonio Cerone and Marco Roveri, editors, *Software Engineering and Formal Methods - SEFM 2017 Collocated Workshops: DataMod, FAACS, MSE, CoSim-CPS, and FOCLASA, Trento, Italy, September 4-5, 2017, Revised Selected Papers*, volume 10729 of *Lecture Notes in Computer Science*, pages 219–229. Springer, 2017. `doi:10.1007/978-3-319-74781-1_16`.

8 Andreea Ancuta Corici, Tina Hühnlein, Detlef Hühnlein, and Olaf Rode. Towards interoperable vaccination certificate services. In Delphine Reinhardt and Tilo Müller, editors, *ARES 2021: The 16th International Conference on Availability, Reliability and Security, Vienna, Austria, August 17-20, 2021*, pages 139:1–139:9. ACM, 2021. `doi:10.1145/3465481.3470035`.

9 Oliver Cvetkovski, Carlo Field, Davide Trinchi, Christof Marti, and Josef Spillner. ZVAX - A Microservice Reference Architecture for Nation-Scale Pandemic Management. International Conference on Microservices (Microservices) – Extended Abstract, may 2022.

10 Oliver Cvetkovski, Carlo Field, Davide Trinchi, and Josef Spillner. ZVAX - Zurich vaccination and pandemic management software. Zenodo, apr 2023. `doi:10.5281/zenodo.7869751`.

11 Oliver Cvetkovski and Josef Spillner. A self-contained, decentralized and nation-scale approach to e-government services. 10th IEEE International Conference on Cloud Computing in Emerging Markets (CCEM) – Student Project Showcase, oct 2021.

12 Mauricio de Vasconcelos Barros, Frederico Schardong, and Ricardo Felipe Custódio. Leveraging self-sovereign identity, blockchain, and zero-knowledge proof to build a privacy-preserving vaccination pass. *CoRR*, abs/2202.09207, 2022. `arXiv:2202.09207`.

13 Anthony Downward, Subeh Chowdhury, and Chapa Jayalath. An investigation of route-choice in integrated public transport networks by risk-averse users. *Public Transp.*, 11(1):89–110, 2019. `doi:10.1007/S12469-019-00194-0`.

14 Yu Gan, Yanqi Zhang, Dailun Cheng, Ankitha Shetty, Priyal Rathi, Nayan Katarki, Ariana Bruno, Justin Hu, Brian Ritchken, Brendon Jackson, Kelvin Hu, Meghna Pancholi, Yuan He, Brett Clancy, Chris Colen, Fukang Wen, Catherine Leung, Siyuan Wang, Leon Zaruvinsky, Mateo Espinosa, Rick Lin, Zhongling Liu, Jake Padilla, and Christina Delimitrou. An open-source benchmark suite for microservices and their hardware-software implications for cloud & edge systems. In Iris Bahar, Maurice Herlihy, Emmett Witchel, and Alvin R. Lebeck, editors, *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS 2019, Providence, RI, USA, April 13-17, 2019*, pages 3–18. ACM, 2019. `doi:10.1145/3297858.3304013`.

15 Andrin Hauri, Kevin Kohler, and Benjamin Scharte. A comparative assessment of mobile device-based multi-hazard warnings: Saving lives through public alerts in europe. *CSS Risk and Resilience Reports*, 2022.

16 Michail-Alexandros Kourtis, Begoña Blanco, Jordi Pérez-Romero, Dimitris Makris, Michael J. McGrath, George Xilouris, Daniele Munaretto, Ruben Solozabal, Aitor Sanchoyerto, Ioannis Giannoulakis, Emmanouil Kafetzakis, Vincenzo Riccobene, Elisa Jimeno, Anastasios Kourtis, Ramon Ferrús, Fidel Liberal, Harilaos Koumaras, Alexandros Kostopoulos, and Ioannis P. Chochliouros. A cloud-enabled small cell architecture in 5g networks for broadcast/multicast services. *IEEE Trans. Broadcast.*, 65(2):414–424, 2019. `doi:10.1109/TBC.2019.2901394`.

17 Peini Liu, Xinjun Mao, Shuai Zhang, and Fu Hou. Towards reference architecture for a multi-layer controlled self-adaptive microservice system. In Óscar Mortágua Pereira, editor, *The 30th International Conference on Software Engineering and Knowledge Engineering, Hotel Pullman, Redwood City, California, USA, July 1-3, 2018*, pages 236–235. KSI Research Inc. and Knowledge Systems Institute Graduate School, 2018. `doi:10.18293/SEKE2018-086`.

18 Manuel Ramírez López and Josef Spillner. Towards quantifiable boundaries for elastic horizontal scaling of microservices. In Ashiq Anjum, Alan Sill, Geoffrey C. Fox, and Yong Chen, editors, *Companion Proceedings of the 10th International Conference on Utility and Cloud Computing, UCC 2017, Austin, TX, USA, December 5-8, 2017*, pages 35–40. ACM, 2017. `doi:10.1145/3147234.3148111`.

19 Sahar Mazloom, Phi Hung Le, Samuel Ranellucci, and S. Dov Gordon. Secure parallel computation on national scale volumes of data. In Srdjan Capkun and Franziska Roesner, editors, *29th USENIX Security Symposium, USENIX Security 2020, August 12-14, 2020*, pages 2487–2504. USENIX Association, 2020. URL: `https://www.usenix.org/conference/usenixsecurity20/presentation/mazloom`.

20 William J. B. Midgley, Michael J. de C. Henshaw, and S. Alshuhri. A systems-engineering approach to nation-scale problems: Municipal solid waste management in saudi arabia. *Syst. Eng.*, 24(6):480–496, 2021. `doi:10.1002/SYS.21597`.

21 Laurie Pickard, Dhawal Shah, and J. J. De Simone. Mapping microcredentials across MOOC platforms. In *Learning With MOOCS, LWMOOCS 2018, Madrid, Spain, September 26-28, 2018*, pages 17–21. IEEE, 2018. `doi:10.1109/LWMOOCS.2018.8534617`.

22 Mohammad Reza Saleh Sedghpour, Aleksandra Obeso Duque, Xuejun Cai, Björn Skubic, Erik Elmroth, Cristian Klein, and Johan Tordsson. Hydragen: A microservice benchmark generator. In *16th IEEE International Conference on Cloud Computing, CLOUD 2023, Chicago, IL, USA, July 2-8, 2023*, pages 189–200. IEEE, 2023. `doi:10.1109/CLOUD60044.2023.00030`.

23 Marwin Shraideh, Patroklos Samaras, Maximilian Schreieck, and Helmut Krcmar. A microservice-based reference architecture for digital platforms in the proteomics domain. In Leona Chandra Kruse, Stefan Seidel, and Geir Inge Hausvik, editors, *The Next Wave of Sociotechnical Design - 16th International Conference on Design Science Research in Information Systems and Technology, DESRIST 2021, Kristiansand, Norway, August 4-6, 2021, Proceedings*, volume 12807 of *Lecture Notes in Computer Science*, pages 260–271. Springer, 2021. `doi:10.1007/978-3-030-82405-1_26`.

24 Hui Song, Phu Hong Nguyen, Franck Chauvel, Jens M. Glattetre, and Thomas Schjerpen. Customizing multi-tenant saas by microservices: A reference architecture. In Elisa Bertino, Carl K. Chang, Peter Chen, Ernesto Damiani, Michael Goul, and Katsunori Oyama, editors, *2019 IEEE International Conference on Web Services, ICWS 2019, Milan, Italy, July 8-13, 2019*, pages 446–448. IEEE, 2019. `doi:10.1109/ICWS.2019.00081`.

25 Sandro Speth, Sarah Stieß, and Steffen Becker. A saga pattern microservice reference architecture for an elastic SLO violation analysis. In *IEEE 19th International Conference on Software Architecture Companion, ICSA Companion 2022, Honolulu, HI, USA, March 12-15, 2022*, pages 116–119. IEEE, 2022. `doi:10.1109/ICSA-C54293.2022.00029`.

26 Akshitha Sriraman and Thomas F. Wenisch. μ suite: A benchmark suite for microservices. In *2018 IEEE International Symposium on Workload Characterization, IISWC 2018, Raleigh, NC, USA, September 30 - October 2, 2018*, pages 1–12. IEEE Computer Society, 2018. `doi:10.1109/IISWC.2018.8573515`.

27 Takanori Ueda, Takuya Nakaike, and Moriyoshi Ohara. Workload characterization for microservices. In *2016 IEEE International Symposium on Workload Characterization, IISWC 2016, Providence, RI, USA, September 25-27, 2016*, pages 85–94. IEEE Computer Society, 2016. `doi:10.1109/IISWC.2016.7581269`.

28 Matthias Vianden, Horst Lichter, and Andreas Steffens. Experience on a microservice-based reference architecture for measurement systems. In Sungdeok (Steve) Cha, Yann-Gaël Guéhéneuc, and Gihwon Kwon, editors, *21st Asia-Pacific Software Engineering Conference,* *APSEC 2014, Jeju, South Korea, December 1-4, 2014. Volume 1: Research Papers*, pages 183–190. IEEE Computer Society, 2014. `doi:10.1109/APSEC.2014.37`.

29 Yanze Wang, Shanshan Li, Huikun Liu, He Zhang, and Bo Pan. A reference architecture for blockchain-based traceability systems using domain-driven design and microservices. *CoRR*, abs/2302.06184, 2023. `doi:10.48550/ARXIV.2302.06184`.

30 Li Wu, Johan Tordsson, Erik Elmroth, and Odej Kao. Microrca: Root cause localization of performance issues in microservices. In *NOMS 2020-2020 IEEE/IFIP Network Operations and Management Symposium*, pages 1–9. IEEE, 2020. `doi:10.1109/NOMS47738.2020.9110353`.

31 Olaf Zimmermann, Mirko Stocker, Daniel Lubke, Uwe Zdun, and Cesare Pautasso. *Patterns for API design: simplifying integration with loosely coupled message exchanges.* Addison-Wesley Professional, 2022.