A full software stack for epidemic disease management: Unlocking the joint potential of software technology and supercomputing

JONAS GILG*, Institute of Software Technology, German Aerospace Center, Germany
JOHANN FREDRIK JADEBECK*, Institute of Bio- and Geosciences, IBG-1: Biotechnology, Forschungszentrum Jülich GmbH, Germany and Computational Systems Biotechnology (AVT.CSB), RWTH Aachen University, Germany
MARIAMA JAITEH*, Helmholtz Centre for Infection Research, Germany
DAVID KERKMANN*, Helmholtz Centre for Infection Research, Germany
NIKLAS MEDINGER*, CISPA Helmholtz Center for Information Security, Germany
SHAHBAZ MEMON*, Jülich Supercomputing Centre, Forschungszentrum Jülich GmbH, Germany
ANNA WENDLER*, Institute of Software Technology, German Aerospace Center, Germany
MORITZ ZEUMER*, Institute of Software Technology, German Aerospace Center, Germany
HENRIK ZUNKER*, Institute of Software Technology, German Aerospace Center, Germany
MAXIMILIAN BETZ, Institute of Climate and Energy Systems, ICE-1: Energy Systems Engineering, Forschungszentrum Jülich GmbH, Germany
RALF HANNEMANN-TAMAS, Institute of Climate and Energy Systems, ICE-1: Energy Systems Engineering, Forschungszentrum Jülich GmbH, Germany
JONAS IMMANUEL HEINICKE, Helmholtz Centre for Infection Research, Germany
JULIAN LITZ, Institute of Climate and Energy Systems, ICE-1: Energy Systems Engineering, Forschungszentrum Jülich GmbH, Germany
ACHIM BASERMANN, Institute of Software Technology, German Aerospace Center, Cologne, Germany
CAS CREMERS, CISPA Helmholtz Center for Information Security, Germany
MANUEL DAHMEN, Institute of Climate and Energy Systems, ICE-1: Energy Systems Engineering, Forschungszentrum Jülich GmbH, Germany
ANDREAS GERNDT, Institute of Software Technology, German Aerospace Center, Brunswick and Center for Industrial Mathematics, University of Bremen, Germany
JENS HENRIK GÖBBERT, Jülich Supercomputing Centre, Forschungszentrum Jülich GmbH, Germany
BJÖRN HAGEMEIER, Jülich Supercomputing Centre, Forschungszentrum Jülich GmbH, Germany
CAROLINA J. KLETT-TAMMEN, Helmholtz Centre for Infection Research, Germany
BERIT LANGE, Helmholtz Centre for Infection Research, Germany
KATHARINA NÖH, Institute of Bio- and Geosciences, IBG-1: Biotechnology, Forschungszentrum Jülich GmbH, Germany
SARAH STRASSBURGER†, Helmholtz Centre for Infection Research, Germany
MICHAEL MEYER-HERMANN†, Department of Systems Immunology and Braunschweig Integrated Centre of Systems Biology, Helmholtz Centre for Infection Research, Braunschweig; Institute for Biochemistry, Biotechnology and Bioinformatics, Technische Universität Braunschweig, Braunschweig; and Lower Saxony Center for Artificial Intelligence and Causal Methods in Medicine (CAIMed), Hannover., Germany
MARTIN J. KÜHN†, Institute of Software Technology, German Aerospace Center, Cologne; and Bonn Center for Mathematical Life Sciences and Life and Medical Sciences Institute, University of Bonn, Germany

Infectious diseases remain a major threat to human societies. During the recent COVID-19 pandemic, mathematical modeling and extensive computer simulations proved highly effective in supporting public health experts and decision makers.

Despite these advances, the full potential of modern modeling approaches and digital technologies has not yet been realized. Many critical tasks – including expert consultations, model execution, scenario analyses, report preparation, and result communication – still relied heavily on manual, human-driven processes with each manual interaction introducing avoidable delays and limiting responsiveness during rapidly evolving outbreaks.

Pandemic preparedness should opt for automated workflows and seamlessly integrated software modules that can improve pandemic mitigation capabilities by substantially reducing response times. For this step, we require robust and flexible computational infrastructure capable of supporting heterogeneous hardware and continuously evolving infectious-disease models. In addition, data sources need to be dynamically integrated. Managing such demands needs infrastructure that supports automated high-performance computing (HPC) workflows. Beyond computational performance, software infrastructure must ensure secure user and data management to comply with data-protection regulations and provide clear, transparent presentation of results to both decision makers and the public.

---

*Shared first authors. These authors contributed equally to this work.
†Shared corresponding authors.

Authors' Contact Information: Jonas Gilg, jonas.gilg@dlr.de, Institute of Software Technology, German Aerospace Center, Brunswick, Germany; Johann Fredrik Jadebeck, j.jadebeck@fz-juelich.de, Institute of Bio- and Geosciences, IBG-1: Biotechnology, Forschungszentrum Jülich GmbH, Jülich, Germany and Computational Systems Biotechnology (AVT.CSB), RWTH Aachen University, Aachen, Germany; Mariama Jaiteh, mariama.jaiteh@helmholtz-hzi.de, Helmholtz Centre for Infection Research, Brunswick, Lower Saxony, Germany; David Kerkmann, david.kerkmann@theoretical-biology.de, Helmholtz Centre for Infection Research, Brunswick, Lower Saxony, Germany; Niklas Medinger, niklas.medinger@cispa.de, CISPA Helmholtz Center for Information Security, Saarbrücken, Germany; Shahbaz Memon, m.memon@fz-juelich.de, Jülich Supercomputing Centre, Forschungszentrum Jülich GmbH, Jülich, Germany; Anna Wendler, anna.wendler@dlr.de, Institute of Software Technology, German Aerospace Center, Cologne, Germany; Moritz Zeumer, moritz.zeumer@dlr.de, Institute of Software Technology, German Aerospace Center, Brunswick, Germany; Henrik Zunker, henrik.zunker@dlr.de, Institute of Software Technology, German Aerospace Center, Cologne, Germany; Maximilian Betz, m.betz@fz-juelich.de, Institute of Climate and Energy Systems, ICE-1: Energy Systems Engineering, Forschungszentrum Jülich GmbH, Jülich, Germany; Ralf Hannemann-Tamas, ralf.ht@gmail.com, Institute of Climate and Energy Systems, ICE-1: Energy Systems Engineering, Forschungszentrum Jülich GmbH, Jülich, Germany; Jonas Immanuel Heinicke, Helmholtz Centre for Infection Research, Brunswick, Lower Saxony, Germany; Julian Litz, julian.litz@icloud.com, Institute of Climate and Energy Systems, ICE-1: Energy Systems Engineering, Forschungszentrum Jülich GmbH, Jülich, Germany; Achim Basermann, Institute of Software Technology, German Aerospace Center, Cologne, Cologne, Germany, achim.basermann@dlr.de; Cas Cremers, CISPA Helmholtz Center for Information Security, Saarbrücken, Germany, cremers@cispa.de; Manuel Dahmen, m.dahmen@fz-juelich.de, Institute of Climate and Energy Systems, ICE-1: Energy Systems Engineering, Forschungszentrum Jülich GmbH, Jülich, Germany; Andreas Gerndt, andreas.gerndt@dlr.de, Institute of Software Technology, German Aerospace Center, Brunswick and Center for Industrial Mathematics, University of Bremen, Bremen, Germany; Jens Henrik Göbbert, j.goebbert@fz-juelich.de, Jülich Supercomputing Centre, Forschungszentrum Jülich GmbH, Jülich, Germany; Björn Hagemeier, b.hagemeier@fz-juelich.de, Jülich Supercomputing Centre, Forschungszentrum Jülich GmbH, Jülich, Germany; Carolina J. Klett-Tammen, Helmholtz Centre for Infection Research, Braunschweig, Germany, Carolina.Klett-Tammen@helmholtz-hzi.de; Berit Lange, Helmholtz Centre for Infection Research, Braunschweig, Germany, Berit.Lange@helmholtz-hzi.de; Katharina Nöh, k.noeh@fz-juelich.de, Institute of Bio- and Geosciences, IBG-1: Biotechnology, Forschungszentrum Jülich GmbH, Jülich, Germany; Sarah Straßburger, Helmholtz Centre for Infection Research, Braunschweig, Germany, sarah.strassburger@helmholtz-hzi.de; Michael Meyer-Hermann, Department of Systems Immunology and Braunschweig Integrated Centre of Systems Biology, Helmholtz Centre for Infection Research, Braunschweig; and Institute for Biochemistry, Biotechnology and Bioinformatics, Technische Universität Braunschweig, Braunschweig; and and Lower Saxony Center for Artificial Intelligence and Causal Methods in Medicine (CAIMed), Hannover., Germany, mmh@theoretical-biology.de; Martin J. Kühn, Institute of Software Technology, German Aerospace Center, Cologne; and and Bonn Center for Mathematical Life Sciences and Life and Medical Sciences Institute, University of Bonn, Bonn, Germany, martin.kuehn@dlr.de.

Meeting the aforementioned challenges requires tight integration of state-of-the-art scientific software with modern, scalable infrastructure that can leverage supercomputing resources when necessary. For rapid deployment in future epidemic or pandemic scenarios, adherence to the FAIR principles for research software is critical to ensure reusability and sustainability.

Keywords: Software platform, High-Performance Computing, Epidemic management, Automation, Pipelines, MEmilio, ESID

1 Introduction

Human societies are largely impacted as soon as a novel major outbreak of infectious diseases occurs. Mathematical modeling and computer simulations have been highly effective in supporting public health experts and decision makers in the recent COVID-19 pandemic [20]. Furthermore, automated data pipelines [31] and digital contact tracing [21] were instrumental in supporting the timely response to and mitigation of COVID-19 spread.

However, the vast potential of today's models and technology have not yet been fully utilized. Requests for expertise, predictions and scenario analyses, preparation of reports, communication of results, or the provision of model simulations were often based on human-to-human interaction and manual interactions with software components. While the generated knowledge already drastically improved the capabilities of decision makers and public health experts to fight COVID-19, substantial and unnecessary delays are the result of each manual interaction. To further enhance pandemic preparedness, robust and flexible computational infrastructure is required across both heterogeneous hardware and evolving infectious-disease models. For instance, during the COVID-19 pandemic, novel data was dynamically integrated and complex expert models were continuously developed [4, 9, 11, 30, 34, 35, 49, 53, 62, 67, 69, 75]. To handle computationally costly models, any infrastructure also needs to support automated high-performance computing (HPC). The targeted infrastructure must also provide robust user and data management to ensure compliance with data-protection regulations as well as clear presentation of results to both decision makers and the public. Meeting these requirements necessitates tight integration of state-of-the-art scientific software with modern, robust infrastructure that leverages supercomputing resources where required. For rapid deployment in future epidemic or pandemic settings, reproducibility and re-usability are key and it is thus important to closely align to the FAIR principles for research software (FAIR4RS) [5].

In this paper, we introduce our open-source software stack for epidemic data integration, modeling, and interactive feedback to support public health decision making and information dissemination during epidemic times. The platform is the culmination of an interdisciplinary project consortium that combines expertise from various domains such as epidemiology, immunology, biology, public health, mathematics, computer science, high-performance computing, and visualization. Using Germany as a case study, we conducted a multi-year pilot phase of our platform, demonstrating the versatility and utility for epidemic mitigation and evaluating how it would have supported the daily work of local health authorities (LHA). With all software components integrated, the platform provides decision makers and public health experts with (immediate) feedback on requests for situation assessments and alternative scenarios, visualizing the results based on daily integrated data and also enabling the automatic integration of model updates into workflows. With the open-source provision of the elementary software with open licenses, our software stack is easily reusable, in its entirety or in parts, and extensible for upcoming epidemic challenges.

Our paper is structured as follows. In Section 2, we provide a short overview of existing software frameworks, stacks, and platforms with respect to pandemic mitigation for public health experts and decision makers. In Section 3, we provide detailed insights into our platform with respect to (i) data sources (Section 3.1), (ii) mathematical modeling approaches (Section 3.2), (iii) data integration

(Section 3.3), (iv) front-end and REST API (Section 3.4), (v) authentication and user management (Section 3.5), (vi) workflow orchestration (Section 3.6), and eventually (vii) the connection to high-performance computing and its infrastructure (Section 3.7). In the results section, we present (i) the platform itself (Section 4.1), (ii) a proof-of-concept with synthetic local data integration (Section 4.2), (iii) the results of testing and user experience (Section 4.4„ and (iv) timings and performance measures (Section 4.3). Finally, we conclude in Section 5.

2 Related work

In particular with the COVID-19 pandemic, a lot of models and modeling frameworks have been proposed [4, 9, 30, 35, 49, 62, 67]. However, these alone represent only one core element of the software stack, as presented in this paper. Based on our search on literature and software and to the best of our knowledge, there is no fully integrated and automated platform for pandemic response.

Analyzing the existing landscape, insights were drawn from a prospectively registered scoping review (Open Science Framework; OSF ID: q32kg), which systematically maps global efforts to develop infectious disease forecasting tools for public-health decision-making [22].

A substantial proportion of identified tools cover only isolated elements of a forecasting workflow, most commonly in the form of static dashboards or disease-specific visualizations without scope for regional adaptation or scenario exploration [19, 29, 72]. Additionally, several platforms provide model outputs without interactive parameterization or integration of contextual inputs, limiting their suitability for operational decision-making in local or subnational public-health settings [17]. Furthermore, many previously described tools are discontinued, not maintained, or no longer accessible following the termination of project-based funding [26, 51, 56], constraining their utility for sustained preparedness or routine public-health operations.

These findings underscore relevant challenges in translating forecasting research into practice and align with the design objectives of platforms such as presented here. In particular, our initiatives focus on regionally adaptable forecasting capabilities, the integration of locally contextualized inputs, and the provision of an interactive, user-oriented visual analytics environment intended to support real-world decision-making. By combining flexible modeling components, locally relevant data structures, and operational usability within a single platform architecture, we seek to address the fragmented landscape identified in the scoping review and to deliver a forecasting solution that is implementation-ready. A comprehensive synthesis of the scoping review's findings will be reported in a dedicated publication [22].

3 Approach and methods

The target users of our platform, ranging from LHAs to decision makers and the general public need to have swift and clear access to ongoing epidemic dynamics with potential outcomes for alternative scenarios to better manage or understand mitigation efforts. First, this creates the need for advanced mathematical modeling that is based on sound data and data integration. Secondly, we need well defined software interfaces, automated workflows, and HPC integration to realize a swift and seamlessly integrated software stack. This section outlines the details of data sources and data integration together with the methodological foundation of our platform, covering modeling and computation, data pipelines, visualization, and communication of results to decision makers.

3.1 Data sources

Reliable data is a key element for mitigating infectious diseases. In the following, we describe the data sources available in the recent COVID-19 pandemic and which we expect to be available in future pandemic contexts. While often the type of data is discussed, the (potential) stratification of the data is also a key element discussed and considered here. As the last meta-dimension of data,

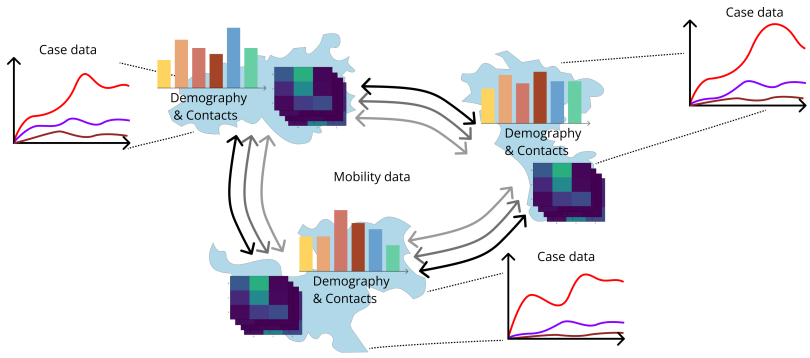

Fig. 1. **A schematic overview of spatially and demographically resolved data for epidemic mitigation.**

we expect data to be provided on a daily basis and with minimal delay to act with a most up-to-date information basis. For a schematic view of the data; see Fig. 1.

In an epidemic setting, we expect to obtain daily reported numbers of positive confirmed, severe, and critical cases, i.e., hospitalized and intensive care cases, as well as deaths. Since a novel pathogen such as SARS-CoV-2 is unlikely to pose an equal threat to individuals of all age groups, the data should be stratified by age. For our SARS-CoV-2 demonstrator case in Germany, confirmed cases and deaths were reported with six age groups of 0-4, 5-14, 15-34, 35-39, 60-79, and 80+ years old [39], and hospitalizations and intensive care numbers without age stratification [38].

Eventually, the spread of SARS-CoV-2 has often been heterogeneous on a spatial scale and data was reported on district level. For the 400 districts we then included demographic data [18] that was aggregated on the age groups specified above. To accurately address spatiotemporal heterogeneity, our models (see Section 3.2) require mobility data or an estimate of regular mobility exchanges between all districts. Here, we used data extrapolated from [12] as well as a data set for inter-district commuting based on a macroscopic transport model and a synthetic population for Germany [71]. In order to quantify the daily number of contacts, we used [48] and its projections [54], split up into the categories “Home”, “School”, “Work”, and “Other”. For the school contacts, we additionally combined [25].

During the first vaccination phases in 2021 and 2022, we additionally used the database [37]. As data was reported with the location of vaccination, we approximated the local vaccination ratios as given by [40]. However, since no vaccination registry is available for Germany, immunity data could not be estimated once a substantial part of the population had recovered or was vaccinated a third or fourth time. In later phases [75], three different immunity layers were assessed through seropositivity studies [42].

General parameters such as the probability of infection, the time to stay infected, the proportion of asymptomatic, severe, and critical infections were quantified from [40, 41] and the references therein.

While the case data described above was provided with short delays, our platform also integrates an option, for each LHA, to supplement the public data sources with own, local data sets. These local data sets may contain additional information regarding the individual vaccination history or virus variant to allow for more precise initialization of models.

3.2 Mathematical modeling

Our modeling components build on the MEmilio framework [9, 10] which is not tied to a single disease, and where models can be used or easily adapted for different respiratory diseases using already existing structures. MEmilio supports models based on ordinary differential equations (ODEs) which can be extended to metapopulation settings [41, 74, 75]. In addition, it allows for equation-based models using Erlang or Gamma [53] or even arbitrarily distributed disease stage transition times [69]. Furthermore, agent-based models (ABMs) [34] or hybrid agent-metapopulation-based [11] models are available. In this work, we focus on the metapopulation formulations and use the ABM for our proof-of-concept data generation process.

3.2.1 Requirements. To capture the dynamics of a specific pathogen, it is crucial to set up a model structure that accounts for the most important aspects of the corresponding disease. Therefore, we identified several general and SARS-CoV-2 specific requirements for a model inside the software stack. These are

- the consideration of asymptomatic development and a presymptomatic infectious phase,
- the inclusion of age-dependent vulnerability,
- the inclusion of heterogeneous immunity and vaccination,
- the inclusion of severe, critical, and deceased states,
- the inclusion of spatial resolution,
- the possibility to compute different scenarios,
- the efficient and optimized implementation of the model

to overall allow public health experts and LHAs study different aspects and local developments of disease dynamics.

3.2.2 Modeling approach. To address the aforementioned requirements, we used and tested an age-resolved *SECIR*-type metapopulation model with two levels of vaccination, denoted *SECIRVVS*; see Fig. 2 for a flow chart between the disease states and [40] for more details.

To make this paper self-contained, we provide a minimal introduction of the SECIRVVS model used throughout our pilot development and testing phase. The model allows for age groups $i = 1, \dots, n_A$, where we chose $n_A = 6$ according to the provided data sources. Furthermore, we have spatial stratification with $k = 1, \dots, n_S$, which was set to $n_S = 400$ to model Germany's 400 districts, and three vaccination or immunity layers $M \in \{\mathrm{N}, \mathrm{PV}, \mathrm{V}\}$. The total number of persons in age group $i \in \{1, \dots, n_A\}$ and region $k \in \{1, \dots, n_S\}$ is given by $N_i^{(k)}$ and similarly, we have the local number of individuals in the particular compartments by $S_i^{(k)}$, $E_i^{(k)}$, et cetera. The (time–dependent) contact matrix $\Phi^{(k)}(t)$ is given by its entries of intra- or inter-age group contact rates by $\phi_{i,j}^{(k)}(t)$, $i, j \in \{1, \dots, n_A\}$. Furthermore, we include isolation and quarantining in our model. Then, the fraction of non-isolated carriers and symptomatic infected individuals is given by $\xi_{I_{NS},j}^{(k)}(t)$ and $\xi_{I_{Sy},j}^{(k)}(t)$, respectively. The force of infection for immunity layer M reads

$$\lambda_{M,i}^{(k)}(t) = \rho_{M,i}^{(k)}(t) \sum_j \phi_{i,j}^{(k)}(t) \frac{\xi_{I_{NS},j}^{(k)}(t) \sum_{M'} I_{NS,M',j}^{(k)}(t) + \xi_{I_{Sy},j}^{(k)}(t) \sum_{M'} I_{Sy,M',j}^{(k)}(t)}{N_j^{(k)}(t) - \sum_{M'} D_{M',j}^{(k)}(t)}. \tag{1}$$

Here $\rho_{M,i}^{(k)}(t) = s_{\kappa_S}(t)\, \rho_{0,M,i}^{(k)}$ captures baseline transmissibility $\rho_{0,M,i}^{(k)}$ and seasonality $s_{\kappa_S}(t) = 1 + \kappa_S \sin\big(\pi(t/182.5 + 1/2)\big)$, where κ_S is a seasonality constant that varied around 0.2. While variant shares can be incorporated multiplicatively, variant regime change was not the focus of this model. For more details; see [40].

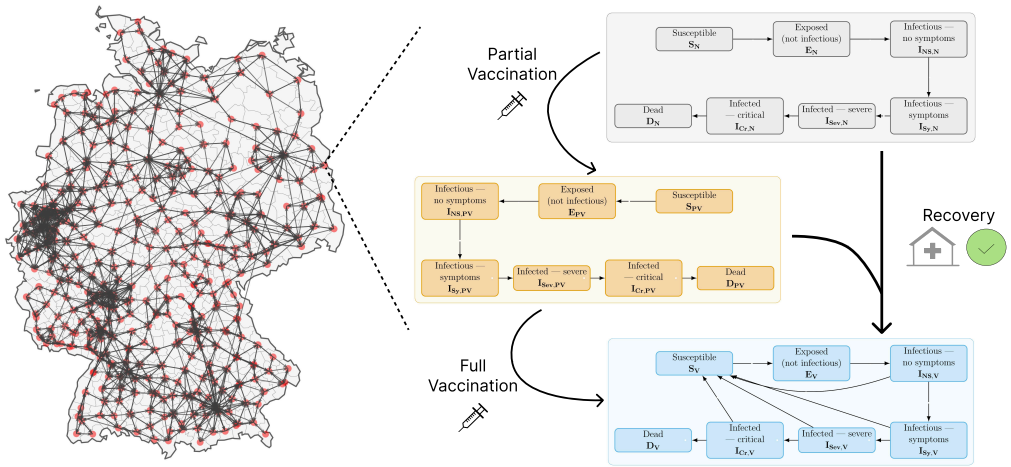

Fig. 2. **Metapopulation coupling and disease progression in the *SECIRVVS* model.** The left panel shows all 400 districts of Germany, with the centroid of each district represented by a red dot. The local models are connected with edges representing mobility. Based on commuter information from [12], we visualize the 2 500 strongest commuter links. Each district contains its own *SECIRVVS* model. As shown in the flow chart on the right, each model has three immunity layers and is stratified in the compartments susceptible (S), exposed/non-infectious (E), infectious without symptoms (I_{NS}), infectious with symptoms (I_{Sy}), hospitalized (I_{Sev}), intensive care (I_{Cr}), and dead (D). Arrows indicate transitions and recovery moves individuals to better-protected layers. Vaccinations are applied as a daily discrete step between susceptible compartments.

Then, the dynamics in the susceptible compartment of layer M can be described by

$$\frac{\mathrm{d}}{\mathrm{d}t} S^{(k)}_{M,i}(t) = -\lambda^{(k)}_{M,i}(t)\, S^{(k)}_{M,i}(t). \tag{2}$$

Other transitions can be defined in a more generic way. Let $z^{(k)}_{q,M,i}$ denote the q-th infection state of our model for age group i and region k and immunity layer $M \in \{\mathrm{N}, \mathrm{PV}, \mathrm{V}\}$. Using mean sojourn times $T^{(k)}_{z_{q,N,i}}$ and transition probabilities $\mu^{(k)}_{z_{q+1,N,i}\,;\,z_{q,N,i}}$ to transit from (naive immunity) state $z_{q,N,i}$ to $z_{q+1,N,i}$, the dynamics in an infection state can be described by

$$\frac{\mathrm{d}}{\mathrm{d}t} z^{(k)}_{q+1,N,i}(t) = \frac{\mu^{(k)}_{z_{q+1,N,i}\,;\,z_{q,N,i}}}{T^{(k)}_{z_{q,N,i}}}\, z^{(k)}_{q,N,i}(t) \;-\; \frac{z^{(k)}_{q+1,N,i}(t)}{T^{(k)}_{z_{q+1,N,i}}}. \tag{3}$$

Individuals with recovery from non-symptomatic, symptomatic, severe, and critical developments are routed to the better protected immunity layers and immunity layer $M \in \{PV, V\}$ modifies both risks and durations. Let $\mu^{(k)}_{\cdot,N,i;\cdot,N,i}$ be the transition probabilities for naive individuals. For risks, we use the naive transition probabilities and add reduction factors p such that

$$\mu^{(k)}_{z_{q+1},M,i\,;\,z_q,M,i} = \frac{p_{z_{q+1},M,i}}{p_{z_q,M,i}}\, \mu^{(k)}_{z_{q+1},N,i\,;\,z_q,N,i} \tag{4}$$

with parameters $p_{E,M,i}, p_{I,M,i}, p_{H,M,i}, p_{U,M,i}, p_{D,M,i}$ representing the age-specific improved immunity due to partial or full vaccination (and prior infection) to reflect better protection against any infection, symptoms, hospitalization, intensive care treatment, and death. Mild infectious periods

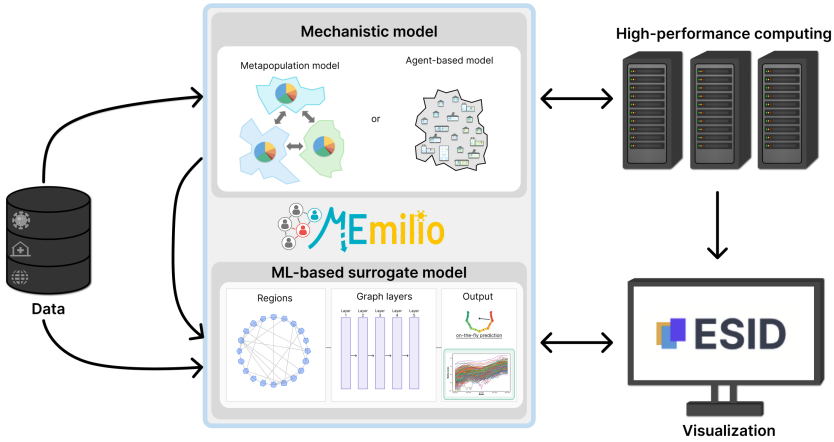

Fig. 3. **Overview of mathematical models and their position within the pipeline.** MEmilio provides mechanistic models such as the graph-based SECIRVVS metapopulation model. As alternative, an ABM could be chosen for integration. Furthermore, an ML surrogate model trained on data generated by the metapopulation models allows on-the-fly simulations and could be integrated into the web front-end (see Section 3.4). Based on available data (see Sections 3.1 and 3.3), simulations are performed, if possible using high-performace computing resources for fast and efficient computation (see Sections 3.6 and 3.7).

in protected layers are shortened via $T^{(k)}_{z_q,M,i} = \kappa_M T^{(k)}_{z_q,N,i}$ with $\kappa_M \in (0, 1]$. Non-pharmaceutical interventions (NPIs) are realized through a change in the contact patterns Φ or through a reduction of mobility on the edges of the graph. Vaccinations are treated outside the dynamical system formulation as a daily update between susceptible compartments of different layers. Recovered individuals are routed to the best protected immunity layer.

To capture spatial dynamics, we embed the local *SECIRVVS* models in a metapopulation approach [41, 75] where each district is represented by its own local *SECIRVVS* model. The regions are coupled via mobility edges that represent commuting and traveling activities between regions. This allows us to model the spatial spread of infections due to mobility between regions. In Fig. 2, we visualize the 2 500 strongest commuter links between the districts of Germany.

Beyond the previous description, MEmilio's modular structure allows to flexibly adapt the model complexity to the specific situation in order to meet the requirements for modeling different pathogens, strains, or regions. The model from [40] as described above has been specifically designed for COVID-19 in Germany when detailed data on age-stratified cases, ICU occupancy and vaccination data was available. It was thus chosen as the state-of-the-art demonstrator in the described software stack.

Although ODE systems can be solved efficiently within the MEmilio framework, due to the resolution on district level and the inclusion of mobility exchange, computation cannot be done on-the-fly – in particular when a wide range of scenarios is considered. To support on-the-fly exploration, we provide machine learning (ML) surrogate models trained on curated simulation outcomes and which replace the mechanistic expert model. The approach based on graph neural networks has been described in [61] and allows instantaneous feedback to public health experts while still integrating spatial and age resolution. Deployment and continuous retraining hooks are integrated in the pipeline but not enabled in production yet.

In order to calibrate models on a local, district-scale, we run Bayesian inference using the probabilistic programming framework PyMC [3]. Similar to [16], we use a student's t-distribution

to define a robust likelihood of the observed case data that is robust to outliers and we incorporate regular change points for the contact rates using gradient-free DE Metropolis-Z [65] sampling algorithm. Convergence is checked using the rank-normalized $\hat{r}$ values [66].

3.2.3 Optimal control. Besides the simulation of scenarios, the MEmilio framework allows to leverage optimal control [7] for ODE models to compute optimal intervention schemes, e.g., schedules of NPIs with optimized strengths, or optimized testing or vaccination strategies.

While MEmilio supports optimal control for graph-based metapopulation models, the optimal control problem considered in our automated pipeline (see Section 3.6) models Germany without spatial stratification to reduce practical runtime. It can be written in its continuous-time formulation as

$$\begin{aligned} \min_{\psi_l(t)} \quad & \int_{t_0}^{t_{\max}} \sum_{l \in \mathcal{L}} \psi_l(t)\, dt \\ \text{subject to} \quad & \text{the } \textit{SECIRVVS} \text{ ODE model}, \\ & I(t) \leq I_{max}, \\ & \psi_l(t) \in [0,1] \quad \forall l \in \mathcal{L}, \end{aligned} \tag{5}$$

where $\mathcal{L}$ is the set of considered NPIs, ψ_l denotes the control variable of the l-th NPI, I_{max} is the number of admissible infections at any point in time, and the objective function minimizes the cumulative exposure to NPIs over the optimization horizon $[t_0, t_{\max}]$. The control variable $\psi_l(t)$ modulates the effect of the l-th NPI on the contact patterns $\Phi(t)$, with $\psi_l(t) = 1$ denoting full implementation and $\psi_l(t) = 0$ no intervention effect (with linear scaling in between).

For implementation, the controls $\psi_l(t)$ are discretized as piece-wise constant segments with length of one week, thus allowing for weekly adaptation of NPI strengths, whereas the constraint $I(t) \leq I_{max}$ is evaluated on a daily basis. The optimal control problem is solved with IPOPT [68] using a direct single shooting approach, see, e.g., [60], and automatic differentiation [50].

3.3 Data integration

3.3.1 Requirements. To integrate epidemiological datasets, from public repository and in-house databases, into a modeling framework, several requirements were identified.

- the data ingestion step should run periodically or on-demand by authorized external services,
- data must be stored according to data engineering standards to ensure efficient and reliable retrieval,
- a secure storage and access to private datasets must be implemented,
- the data integration interface must be scalable to enable efficient interaction with large datasets,
- communication and interoperability with other services should be ensured.

3.3.2 General overview of MemiliSQL REST API. The SARS-CoV-2 epidemiological datasets have been continuously gathered by German LHAs and shared with the Robert Koch-Institute (RKI) throughout the the pandemic. The data shared with the RKI was subsequently made available via the RKI public data repositories. To provide the needed datasets (see Section 3.1) on demand to the modeling engine behind the platform, an Extract-Transform-Load (ETL) data pipeline was implemented. The pipeline periodically ingests and transforms data from hospitalization and intensive care units (ICU) [38], vaccination [37], and infection cases [39] from the RKI repositories using the MEmilio epidata package [10] before persisting raw and processed data in a managed datastore. In addition, demographic data from the Federal Agency of Work and the Federal Statistical Office of Germany (DESTATIS) is harvested [18] and stored in the managed datastore.

In addition to data from public repositories, richer data in LHA in-house SurvNet databases [36] can be ingested and integrated in our solution. These datasets can contain features not present in public data. With the support of a pilot LHA, we created an SQL query to extract a set of epidemiological information, e.g., vaccination status, intensive care or general patient information. For data protection reasons, only authenticated LHA employees can run the query and upload the aggregated output CSV file using the upload functionality implemented in ESID; see Fig. 5. The uploaded data can then be sent to a staging location in MinIO File Storage awaiting transformation. To validate this feature, we first used real data provided by the pilot LHAs from times of high pandemic activity. Second, we generated artificial individual pandemic data to prove the functionality of the whole workflow. To generate individual data, we use MEmilio's agent-based model [34]. Various features are used to assess data quality, perform data transformation, upload and extract the datasets into the database. Data security is ensured through our user management system, cf. Section 3.5.

The ETL data pipeline was implemented as a standalone REST API built on FastAPI [47], coined MemiliSQL, and enhanced with Celery [63] and Redis broker [45] as task queue system. The MemiliSQL REST API allows other applications to trigger data pipeline steps. Each ETL data pipeline step, i.e., data ingestion, data transformation, data upload to the MemiliSQL database, and data extraction from the MemiliSQL database, has been defined as an API endpoint and can be requested using the *GET* operation. Moreover, since data extraction and transformation steps are demanding in terms of computing resources and could not be handled by the MemiliSQL REST API, these were implemented as Celery tasks. These tasks can be scheduled to run asynchronously across different Celery workers within the Kubernetes pods dedicated to handle this workload, and their outputs can be temporarily stored in the Redis database. Offloading the REST API with a task queue allowed the API to remain responsive to incoming tasks while also scheduling their runs by workers. To create a fully automated and orchestrated data flow, we have implemented Airflow directed acyclic graphs (DAGs). Each DAG sends one or multiple API requests against the MemiliSQL REST API to create and initiate Celery tasks. The Fig. 4 gives a high-level overview of the ETL services that support the integration of both public and synthetic LHA datasets. A description of the different Airflow DAGs is given in Section 3.6.

3.3.3 Deployment of the data integration services. The architecture and infrastructure hosting the data pipeline is rather minimal. It comprises the MemiliSQL REST API and a datastore, which consists of a PostgreSQL database and MinIO File Storage. The deployment of applications (e.g., MemiliSQL REST API and database) is configured and managed by creating Fleet[1] bundles from Helm charts[2], which are stored in a Git repository. While the community PostgreSQL Helm chart was used for the MemiliSQL database, a tailored Helm chart based on the containerized MemiliSQL REST API was created. Rancher Fleet is subsequently used to automate the deployment of these Fleet bundles as applications on a Kubernetes cluster. In addition to software application bundles, Fleet bundles were defined to create storage applications which provision and attach persistent volumes to Kubernetes database-containing pods. The provisioning of storage requires access to a computing infrastructure, and we used Openstack[3], an Infrastructure-as-a-Service (IaaS) platform, to host the Kubernetes cluster and provision CPU and RAM resources. Fleet continuously synchronizes the Kubernetes resources with the Git repository containing the Fleet bundles. This ensures the reproducibility and versioning of the overall infrastructure. Moreover, the deployment into a

[1] **Rancher Fleet**. A container management and deployment engine that manage GitOps for a single Kubernetes cluster. https://fleet.rancher.io/

[2] **Helm**. A Kubernetes-native package manager for templated and version-controlled application deployments. https://helm.sh/docs/

[3] **OpenStack**. Open-source cloud computing platform for building private and public clouds. https://www.openstack.org

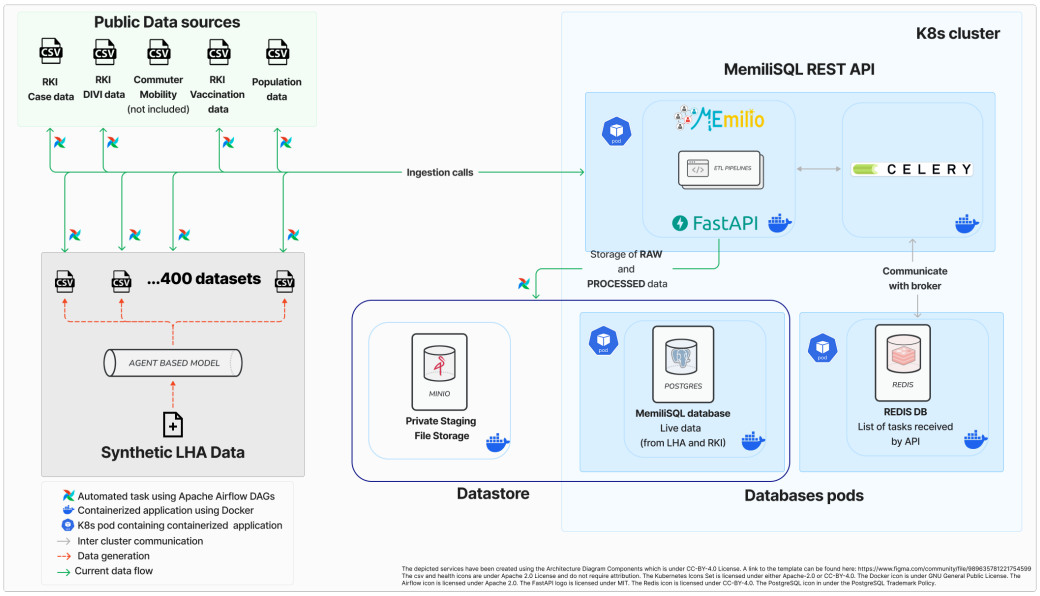

Fig. 4. **A general overview of the data integration services.** The data integration services comprise a datastore and a REST API. Upon receiving requests, the API will proceed with the scheduling of various tasks, including the ingestion of specific dataset, its transformation and finally its upload into the MemiliSQL database.

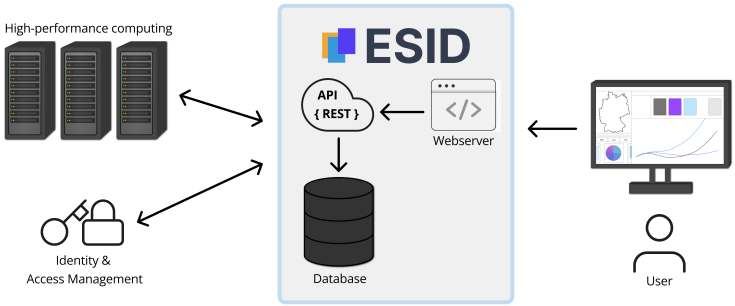

Fig. 5. **ESID's architecture.** ESID's front-end can be accessed by the user and the ESID's back-end is accessed through the user query or an automated instance. The API verifies all actions with the identity and access management system.

Kubernetes cluster enables reliable orchestration, provisioning, and management of computational resources.

Data ingested from public data sources and data uploaded by LHAs are stored in the MemiliSQL database, which is a PostgreSQL database. The amount of storage needed for one day of data from public repositories varies. For one year of data collection, the storage needs to store raw and transformed datasets is estimated to be 500 GB.

3.4 Front-End and REST API

To ensure low-barriers for system access as well as swift and smooth adoption in pandemic situations, the web front-end ESID[4] (*Epidemiological Scenarios for Infectious Diseases*) was developed with a corresponding REST API in the background. Fig. 5 shows the architecture of ESID and the other services it communicates with. The development was closely aligned with the guidelines reported in [8], in particular, addressing multiple user groups such as public health experts, decision makers, and individuals from the general public. By providing access to the general public, the platform fosters transparency and, thus, should increase acceptance for implemented interventions.

3.4.1 Requirements. The development of the user front-end was driven by the idea of a minimal barrier toolkit for operating and interconnecting with epidemiological models. This means that it should not require particular privileges to be installed or operated, and interfaces should be built upon standardized approaches. In line with current development in the public health sector in Germany [14], we therefore opted for a web-based application. Many requirements for a visual analytics toolkit were already discussed in a recent publication [8], and a crucial additional step was the inclusion of four German health authorities that served as pilot LHAs to ensure that the final product was developed according to the users' needs.

We collected more than 100 user stories. From these, a handful of relevant requirements defined the choice of technology and architecture:

- Retrospective and ongoing pandemic data need to be visualized,
- filtering in all dimensions of the dataset should be possible,
- interface response should be immediate, if possible,
- multiple user groups with different rights should be implemented,
- selected users should be able to trigger simulations with custom parameters,
- selected users should be able to upload data to be included in simulations.

3.4.2 REST API. The ESID API[5] serves as the critical middleware layer connecting the simulation back-end with the web-based front-end, facilitating seamless data flow and management within the system.

Built on FastAPI [47], the API provides a robust RESTful interface coupled with a PostgreSQL [27] database for persistent storage and efficient data retrieval. Additionally, the API uses Celery web workers [63] for more complex tasks, and stores temporary task results in an in-memory database using Redis [45]. To optimize query performance, we employ strategic indexing of frequently accessed fields, ensuring rapid data retrieval even as the volume of simulation results grows. The API exposes two distinct categories of endpoints, each serving different architectural needs. back-end-facing endpoints handle model definition and data ingestion, facilitating integration with MEmilio (see Section 3.2) and Apache Airflow used for workflow orchestration (see Section 3.6). These endpoints enable automated workflows to receive simulation requests, run simulations, and store results. Front-end-facing endpoints, conversely, provide optimized data access for visualization purposes, returning appropriately aggregated and filtered datasets based on user interaction, and submitting new simulations for approved users. The following endpoints exist to manage the complex and dynamic requirements:

- **Models** are the foundation of all simulations. They define all model parameters; in which groups to split the population, and in which compartments the infection can be split into.

[4] **ESID Front-end**. The React-based website for ESID. https://github.com/DLR-SC/ESID
[5] **ESID Back-end**. The back-end for the ESID website based on FastAPI and PostgreSQL. https://github.com/SciCompMod/ESID-back-end

- **ParameterDefinitions** define the rules of the model, e.g., how likely it is for a person to become infected, how long the infection lasts, or how severe infections can get.
- **Groups** represent a flexible concept to stratify by age, gender, or socio-economic properties.
  - **Categories** specify which groups belong together, e.g., the age category tracks all groups which specify an age range. Importantly, each person belongs to exactly one group of a category.
- **Compartments** represent all the infection states of a model.
- **Nodes** are a generic way to define a spatial subdivision, generally given by administrative units.
  - **NodeLists** group nodes together into a named set.
- **Interventions** define disease mitigation measures each defined through a strictness and application area.
- **Scenarios** are the core of the application and combine the information above. Each scenario represents a simulated prediction, where a model with parameters, a set of nodes, a time frame, and interventions are chosen.
  - **InfectionData** is the endpoint to get and filter the multidimensional data of a scenario. It offers parameters to filter by nodes, dates, compartments, groups, and percentiles.

For all endpoints, at least *GET*, *POST*, and *DELETE* operations exist. The *Scenarios* endpoint has an additional *PUT* operation to upload simulation data.

The authentication system (see Section 3.5) regulates which endpoints can be accessed with which permissions. Generally, all *GET* operations on endpoints are accessible without authentication; although they might only return results tagged for public access. So, even though they are publicly accessible, they return different results depending on who requests the data. *POST*, *PUT*, and *DELETE* operations on endpoints always require authentication. *Models*, *ParameterDefinitions*, *Groups*, *Compartments*, *Nodes*, and *Interventions* can only be managed by system administrators or other authenticated services such as Apache Airflow (see Section 3.6). Only authenticated users can create *scenarios* and can subsequently share access with other users.

3.4.3 Front-End. To fulfill the requirements of a modern and easy to use interface, the front-end was designed as a sophisticated visual filtering system based on brushing and linking principles [33], enabling intuitive exploration of multidimensional simulation data. ESID is built on top of established frameworks, like React [52] and Redux [1], which allows the application to be highly interactive. The architecture is based on VVAFER, proposed by the authors of [73]. The MUI library [59] for interface components provides a modern look following the Material Design guidelines [44]. Fig. 6 shows the main interface of ESID, which contains four major connected components. Each component gives insight into at least one dimension (for example, time, space, and infection states) and actions on one component interactively as a filter and changes the view of the other components.

To make ESID as accessible and user-friendly as possible, several features have been added. On first visit, an interactive on-boarding tour guides users through the platform's capabilities. By highlighting important aspects and encouraging the user to interact with the components, users can quickly familiarize themselves with each component's functionality; see Fig. 7 (left). The interactive tour can always be found in the help center which also provides a natural language based semantic search of related resources to answer users' questions.

A key technical challenge addressed in the front-end is state synchronization between the client application and the API. All selections by the user in the interface can be interpreted as a large filter for a data cube. The front-end maintains a local state that accurately reflects available data and current filters, and uses it to fetch the requested data efficiently. Data communication

Fig. 6. **Main user interface of ESID.** The numbered components each show a slice of the data cube, as well as serving as filters for the other components. 1) The scenario cards show each scenario with data for the selected date, infection state, and node, including a percentage of change that shows the delta compared to the reference date; each can also be expanded for group specific data. Selecting a scenario changes the visualization of other components to show this scenario's data. 2) A list of infection states (potentially grouped from compartments) that show data for the reference day displayed above the list. Selecting an infection state changes the other components to show data for this infection state. 3) A map with regions (nodes) showing the spatial patterns of the selected data. Selecting a region changes the other components to show data for this region. 4) A time series view showing the selected data. Contrary to other components, it can show multiple scenarios at once, to allow easy comparisons. For the selected scenario, additional information is shown with uncertainty bands (e.g., 25th and 75th percentiles) and dashed lines for group specific data. The vertical dashed line represents the reference day and the purple line the selected date. Selecting a day changes the other components to show data for that day. The small gap between the selected scenario (blue line) and the retrospective data (black line) can be explained by the comparably low number of runs conducted here and the stochasticity in the model initialization.

with the back-end is managed through RTK Query [2], which implements intelligent caching strategies and pre-fetching to minimize latency and enhance user experience. These strategies store frequently accessed datasets locally, reducing redundant API calls and enabling instantaneous filtering operations on cached data.

A dedicated scenario creation interface allows authorized users to configure and submit simulations with custom parameters; see Fig. 7 (right). Every creation induces an interaction loop: A newly created scenario gets posted to the REST API. Airflow (see Section 3.6) then submits all new scenarios to MEmilio (see Section 3.2) for simulation and writes the results to the REST API. After simulation of the scenario, the user can evaluate the effects of the contact reduction through the NPIs, drawing individual conclusions and submit further scenarios. Authentication flows in the front-end mirror the API's tiered approach, seamlessly integrating with the identity provider (IDP) presented in Section 3.5.

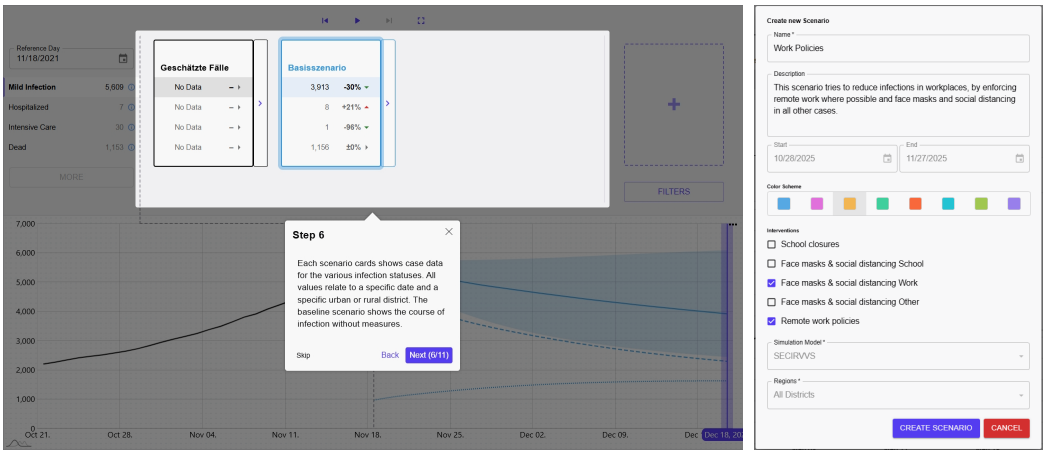

Fig. 7. **Left: Excerpt of the interactive on-boarding of the front-end.** The scenario card view is highlighted and explained; other components are grayed out. **Right: Dialog to create a new scenario.**. The user can configure information for a custom scenario.

3.5 Authentication and authorization system

The public front-end from the previous section allows our users to trigger, visualize, and analyze simulations of infectious disease spread. These simulations might need substantial computational resources and may contain sensitive data that was uploaded by an LHA but is publicly undisclosed. It is therefore crucial that only authorized entities can access specific functionalities. We implement an authentication and authorization system that allows us to cater to a wide range of potential user groups, e.g., the general public, LHA employees, and public health experts.

3.5.1 Requirements. We first identified the need for different user roles and categorized the system's users into three user groups: the general public, IT staff from LHAs, and public health experts from LHAs. The latter then also serve as connection point to decision makers. For the general public no requirements apply and they can freely use the web front-end as a source of information. To manage the user base, the IT staff of LHAs is allowed to administer the local health experts; we summarize the relations between user types in Fig. 8.

We furthermore require a system that allows for the decentralized on-boarding and management of users at different LHAs, fine-grained access control of our sensitive API, and resource sharing as well as collaboration between different LHAs and expert users. Eventually, the system should be open to extension and integration with other identity providers such as the German BundID [23]. To generally address the requirement of a secure authorization and authentication system, standardized and state-of-the-art protocols should be used.

3.5.2 Implementation. To implement our authentication and authorization system, we use the OAuth 2.0 authorization framework [28] and the OpenID Connect authentication protocol [58] together with a self-hosted Keycloak [24] server as our identity provider.

OAuth 2.0. OAuth 2.0 is an authorization framework that allows a third-party application to obtain limited access to a (HTTP) service on behalf of a consenting user without exposing the user's credentials or identity to the application. For instance, a user can grant a printing service access to their personal cloud data, without sharing their credentials with the printing service.

Formally, OAuth 2.0 defines four roles: resource owner, client, authorization server, and resource server. The resource owner is an entity capable of granting access to a protected resource. Usually, this is a person and we refer to them as end-user. The client is an application that makes requests to access protected resources on behalf of the resource owner with their authorization. In practice, the client is often a web-application running in the browser and the end-user authorizes the client by logging into the authorization server. The authorization server issues access tokens to the client after it successfully authenticated the resource owner and obtained authorization. Lastly, the resource server is hosting the protected resources and responds to requests to the protected resources using access tokens. In our use case, the resource owners are the public health experts at LHAs, the client is the web front-end, the authorization server is our self-hosted Keycloak instance, and the resource server is the back-end REST API.

To increase its flexibility, OAuth 2.0 provides so-called flows, each specifying an authorization protocol. Since an in-depth discussion of each flow is out of scope for this work, we refer the reader to its specification [28] for more details, and, instead, focus our attention on the flow we chose for ESID: the Proof Key for Code Exchange (PKCE) flow. Defined in [57], the PKCE flow is an extension of the authorization code flow that protects clients against the interception and the injection of authorization codes by introducing a random code challenge in each request. We chose this flow as it is the recommended flow with the best security guarantees for single page applications like ESID. Fig. 9 shows the login flow of ESID.

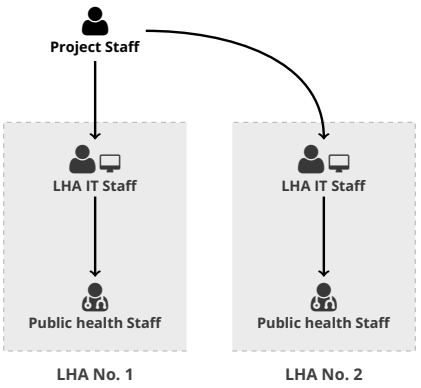

Fig. 8. **The different users of ESID and their relationships.** The project staff administers local IT staff who, in turn, administer their public health experts.

To explain the PKCE flow, we will now focus on the communication between the web front-end (the client) and our Keycloak instance (the authorization server). The goal of the flow is to let the web front-end act on behalf of the end-user using it such that it can, for instance, make calls to APIs that are only available to public health experts. To achieve this, the web front-end will first create a code verifier by generating a random high-entropy string, and a code challenge by hashing the code verifier. Then, the web front-end will request an authorization code from Keycloak and also transmit the code challenge. If the end-user logs in, i.e., authorizes the web front-end, Keycloak will return an authorization code. The authorization code is a unique, short-lived, single-use token that Keycloak associates with the request, in particular, with the code challenge. Next, the web front-end will exchange the authorization code and code verifier for an access token, which it can then use to make requests on behalf of the end-user. To do so, the server will check that the code challenge it associated with the authorization code is really the hash of the code verifier and, if so, respond with an access token. Once the web front-end obtained the access token, it will send it along any requests to the web front-end API which will verify the token using the public key of the Keycloak realm the user is logged into, and, if successful, allow the web front-end to act on behalf of the end-user.

OpenID Connect. OpenID Connect [58] is an identity layer built on top of the OAuth 2.0 protocol that allows a client to verify the identity of an end-user established by authentication with an authorization server. We refrain from formally introducing and outlining how OpenID Connect works but instead only describe how it builds upon the PKCE flow we have chosen.

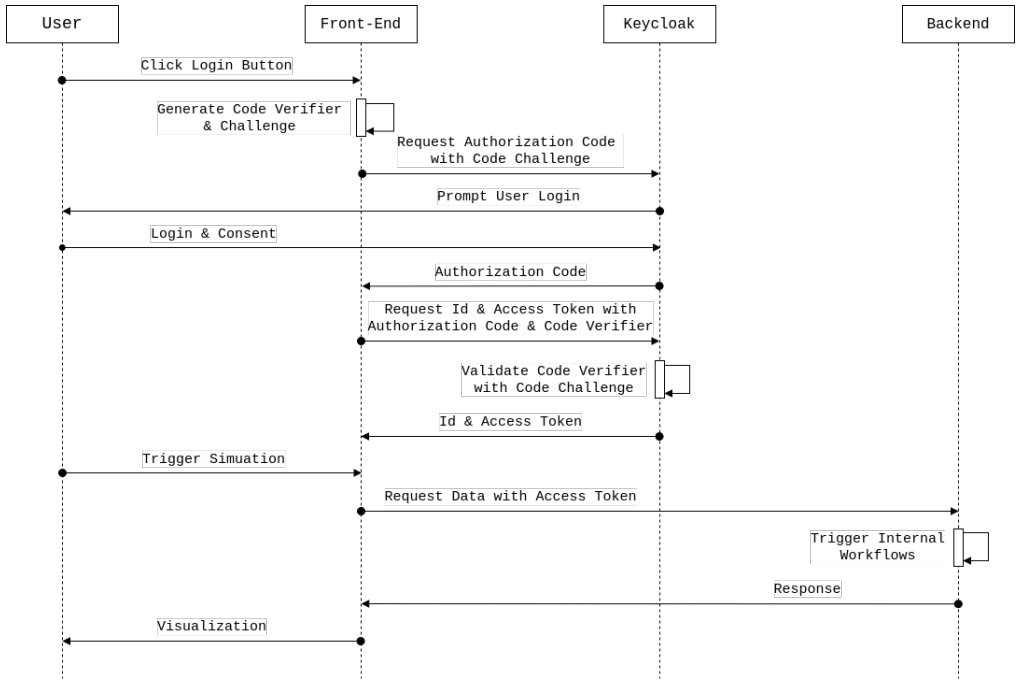

Fig. 9. **Overview of the login flow of ESID.** First, a user logs into ESID by clicking the login button, which starts the PKCE flow between ESID and our Keycloak instance. When ESID exchanges the authorization code and code verifier for the ID token and access token, the flow is finished and ESID can now make requests to the back-end API on behalf of the user potentially triggering internal workflows as described in Section 3.6.

To allow for authentication, OpenID Connect uses an ID token which is issued by the authorization server and that contains information about the end-user and their authentication status with the authorization server, e.g., which realm they are logged into and which roles they have. In the case of the PKCE flow, this token is returned together with the access token when exchanging an authorization code. The token is then sent along with the access token to the REST API which can use it to, for instance, check whether the user is logged in, and to check which LHA the user belongs to.

Keycloak. Keycloak is an open-source identity and access management software supporting standard protocols such as OAuth 2.0 and OpenID Connect, two-factor authentication, as well as permission and role-based access management. In our system, Keycloak possesses the role of the identity provider (Idp) or, to put it in the terms of OAuth 2.0, the authorization server. That is, our users have an account at our Keycloak server, it stores their credentials, and, subsequently, authenticates them for our other services. We self-host the Idp is to ensure ownership of the user's data and to not rely on other Idps which might not operate under EU data protection laws.

To model the relationships between our users as depicted in Fig. 8, we use Keycloak's role-based access management and its concept of *realms*. A realm is a logically isolated identity provider for a single application. That is, each realm holds its own user base, which cannot interact with the users of other realms. The main purpose of realms is to allow for a single Keycloak instance to serve multiple, independent applications. However, we use realms to isolate the users of different LHAs by creating one realm per LHA, achieving our goal of making it impossible for admins of

one LHA to interact with the staff of other LHAs. To give admins the rights to manage the public health experts at their LHA, we use a simple role-based permission system which allows users that have the admin role. To protect our API, we can then check whether a user is logged in with our Keycloak server, which realm (i.e., LHA) they belong to, and whether they have the correct permissions.

3.6 Pipelines and workflow orchestration

To provide decision makers and public health experts with accurate monitoring and time-critical decision support, up-to-date models (see Section 3.2), daily available data (see Sections 3.1 and 3.3), and the accessible front-end (see Section 3.4) need to be supplemented by automatically executed workflows that substantially reduce manual intervention loops. To this end, our platform adopts state-of-the-art workflow orchestration practices using Apache Airflow, ensuring that workflows of multiple steps are run in the right order, at the right time, and on the right hardware. Reliability and reproducibility at scale are achieved through a cloud-native deployment of Apache Airflow on a Kubernetes cluster managed via Rancher, running on an OpenStack-based IaaS layer. This setup provides modular integration of additional hardware and software components as needed. Furthermore, Unity-IDM[6] is used for federated, institution-backed authentication to provide controlled access and fine-grained user management for both development and operation. Finally, MinIO[7] provide workflows with an S3-compatible persistent object storage back-end MinIO, enabling multiple workflows, even running on heterogeneous hardware, to efficiently share and reuse the same datasets.

3.6.1 Requirements. The implementation of our pipeline or workflow is driven by a set of requirements that ensure automation, scalability, robustness, and reliability in managing epidemic mitigation workflows. These requirements cover both functional and nonfunctional aspects, supporting management and processing of data in heterogeneous computational environments. In the following, we provide a high-level overview of the requirements.

- Provide end-to-end automation from data collection to result dissemination, with reproducible, traceable execution without manual intervention,
- support dynamic scalability to accommodate varying computational loads and capability to allow new models, data sources, and sinks to be integrated.
- ensure reliable storage, sharing, and versioning of intermediate and output data in accordance with FAIR (Findable, Accessible, Interoperable, Reusable) principles [70],
- enforce fine-grained, federated authentication and authorization for shared services (e.g. pipeline source code, object storage), based on institute-backed identity assurance, with adequate access, modification, and deployment of workflow resources and stored results to authorized storage back-ends,
- provide continuous operation with acceptable performance, minimal downtime, and resilient to transient failures,
- interoperate with institutional and external infrastructure services, and compatible across containerized environments,
- capture and expose detailed workflow execution metadata to support runtime analysis and auditing.

[6] **Unity-IDM**. Identity and Access Management platform supporting federated authentication and fine-grained user management. https://unity-idm.eu

[7] **MinIO**. High-performance, open-source, S3-compatible object storage system. https://www.min.io

3.6.2 Architectural Realization. The architecture of the LOKI pipeline brings together workflow orchestration and enactment, container execution, data management, and authentication into a unified automation environment to support the visualization of epidemic developments. It is organized into three tiers – the User Domain, the Container (Platform) Domain, and the IaaS Domain – which separate workflow logic from orchestration and infrastructure management (see Fig. 10). This architecture improves maintainability, scalability, and security across distributed environments and extends the foundational concepts introduced by [46].

At its core, Apache Airflow defines workflows as directed acyclic graphs (DAGs) where nodes represent individual tasks and edges define dependencies. The scheduler regularly evaluates all active DAGs, determines tasks whose dependencies have been satisfied, and submits them for execution. The executor provides the bridge between Airflow's orchestration logic and the underlying compute environment. In our deployment, we use the executor of type KubernetesExecutor, which is used to run tasks in a Kubernetes-based environment. This executor dynamically creates a dedicated *Task Pod* for each task. A *Task Pod* is an ephemeral instance in a containerized environment with its own resource limits (CPU and memory), mounted volumes, and isolated runtime context. Upon task completion, the corresponding *Task Pod* is automatically terminated, allowing cluster resources to be released and reused efficiently.

All workflow elements, including DAGs, executors, and configuration templates, are maintained in a central GitLab repository following a workflows-as-code paradigm. Airflow automatically synchronizes these elements via a Git-based deployment mechanism, which allows consistent versioning across development and production environments. The Airflow deployment configuration, such as container image references, runtime parameters, and synchronization policies, is managed declaratively using Helm charts[8] in accordance with Infrastructure-as-Code (IaC) principles [55]. The Helm chart itself is version-controlled in GitLab. While keeping both the workflow code and the deployment configuration in Git, the pipeline and the Airflow deployment can be tracked and reproduced consistently across environments.

The User Domain (see Fig. 10, grey section) is comprised of workflow authors, data scientists, and service accounts interacting with Airflow through its native web interface or REST API. Authentication and authorization are handled by Unity-IDM using OpenID Connect and OAuth 2.0 standards, providing federated and interoperable access control consistent with institutional defined identity policies. Within this layer, the Airflow webserver was extended to support custom role handling and dynamic redirect URI processing during authentication.

The Container Orchestration tier (see Fig. 10, blue section) hosts Airflow components—scheduler, webserver, and executor—alongside dynamically created taskpod instances. The underlying deployment is based on a Rancher-managed Kubernetes cluster, which handles orchestration, scaling, and lifecycle management. As mentioned above, each taskpod is assigned explicit CPU and memory requests and limits to achieve fair workload distribution. Intermediate workflow tasks communicate through the shared MinIO object store instance for data exchange and storage of intermediate or final results.

Workflow metadata including DAG definitions, task states, configurations, and execution logs are maintained within the same platform layer. This provenance metadata is stored in a centralized PostgreSQL database instance providing traceability and auditability of workflow executions. The execution logs are aggregated on a shared Network File System (NFS) volume mounted across Airflow components. Since logs from daily runs can grow rapidly, retention and cleanup policies are applied to prevent storage saturation and help maintaining stability across nodes.

[8] **Helm**. A Kubernetes-native package manager for templated and version-controlled application deployments. Documentation available at: https://helm.sh/docs/.

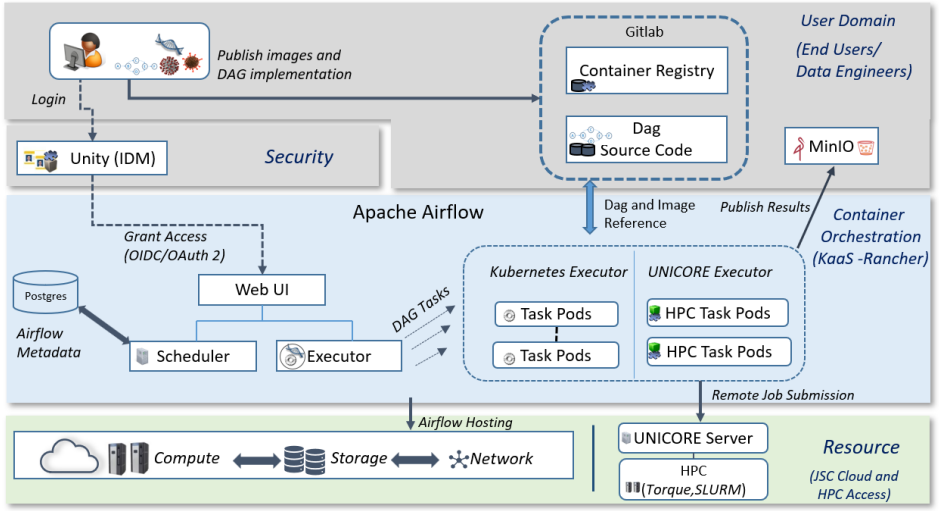

Fig. 10. **Automated workflows architecture**. This figure shows the User domain (grey section), Container Orchestration tier (blue section) and Resources or IaaS tier (green section) which also highlights the integration with the HPC environment via UNICORE.

The Resource tier (see Fig. 10, green section), which provides the underlying infrastructure. This domain relies on an OpenStack-based IaaS, provisioning compute, storage, and networking resources for all Kubernetes nodes. Each Kubernetes node consumes a virtual machine configured via OpenStack flavors that define CPU, RAM, and ephemeral storage. The PostgreSQL database, object store, and NFS server are also hosted within this domain. Depending on the actual capacity of the infrastructure, resource quotas and volume allocations for the Airflow deployment are explicitly defined at this layer. This tier also highlights the connection to HPC resources through UNICORE [64], enabling Airflow to delegate compute-intensive simulation tasks to remote HPC resources whenever their runtime requirements exceed the limits of Kubernetes deployment. Details on the integration of external HPC resources via UNICORE (see Section 3.7).

3.6.3 ESID pipeline. The *ESID pipeline* is a core workflow that automates data management and processing of reported and simulated data. It is composed of several data- and compute-oriented steps, including data collection, model preparation, and scenario-based forecasting. The data ingestion logic and underlying ETL processes are described in detail in Section 3.3.2. Fig. 11 provides a visual depiction of the ESID pipeline, which is structured into three main phases: data ingestion, model preparation, and post-processing with results upload.

The *Data Ingestion* phase consists of four tasks that retrieve heterogeneous epidemiological and demographic datasets from public sources. In practice, reported COVID-19 incidence and case statistics are obtained from the Robert Koch Institute (RKI) through the `get-case-data` task. The `get-divi-data` task retrieves information on ICU occupancy. Vaccination coverage is collected using `get-vaccination-data`. In addition, `get-pop-data` loads population and demographic distributions for German administrative regions. The resulting datasets are harmonized into a unified internal representation for downstream processing.

The subsequent *Model Preparation* phase begins with the test connection task (`test-authentication`), which verifies connectivity to remote compute and storage environments hosted at the Container Orchestration tier (depicted in Fig. 10). The `compute-initialization` task aggregates all inputs

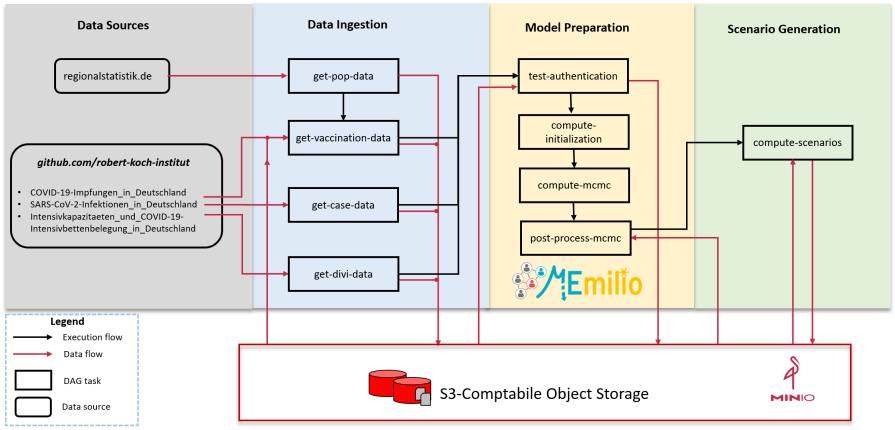

Fig. 11. **ESID Pipeline consisting of three phases.** The figure shows the high-level workflow phases of data ingestion, model preparation and scenario generation including post processing and results upload, as implemented in Apache Airflow. The pipeline integrates data sources, distributed computation, and S3-compatible storage via MinIO.

and constructs the compartmental model structures required for parameter calibration. These structures are then used by the distributed `compute-mcmc` tasks, which perform one global parameter estimation task using Markov Chain Monte Carlo sampling. This phase concludes with the `postprocess-mcmc` task, which is responsible for generating posterior summaries and persisting intermediate results for subsequent analysis.

The *Scenario Generation* phase represents the `compute-scenarios` task, which takes the posterior summaries from the previous phase and simulates epidemic trajectories under varying parametric assumptions. It finally uploads the resulting data to the S3-compatible storage system (MinIO) for visualization and decision support. Furthermore, the task includes an optimal control scenario configured as described in Section 3.2.3, whose optimized control variables are subsequently applied to a simulation with district-level resolution as a basis for results visualization.

The ESID pipeline runs in an Airflow–Kubernetes environment and is subject to several practical constraints that affect scheduling and runtime behavior. Since the pipeline is executed as part of a daily reporting process, each run must finish within a fixed time window. This limits the amount of computation that can be performed per run, even when additional HPC resources are available. Because workflow tasks are executed as individual Kubernetes pods, the level of parallelism depends on the cluster's available CPU, memory, and node capacity. When many tasks run concurrently, this can lead to scheduling delays or pod evictions.

Some pipeline steps also depend on external data sources and institutional authentication services. Variations in availability or response times of these services can delay pipeline startup or execution. In addition, the pipeline is deployed in a distributed setup, where Airflow DAGs and container images are stored in a remote GitLab registry. Pulling updated images from external sources can introduce startup delays, especially when node caches are cold. Finally, repeated execution of containerized tasks can lead to the accumulation of cached image layers and other temporary data on compute nodes. If this data is not cleaned up, it can degrade performance over time.

3.7 Automation combined with High-Performance Computing

For computationally heavy and highly parallelizable workflows, access to distributed HPC resources is essential to produce near-time results, e.g., in daily analyses and short-term prediction of infection case numbers. Achieving this requires a middleware that connects Airflow to HPC systems. In the context of this work, the HPC system of our software stack should interact with JURECA [32], a pre-exascale modular supercomputer operated by the Forschungszentrum Jülich. JURECA offers several types of compute nodes. Daily analyses, inference, and short-term prediction are all tasks suitable for the standard compute nodes offered by JURECA. Each standard compute node offers two AMD EPYC 7742 processors (2×64 cores running at 2.25 GHz) and 512 GB of DDR4 memory.

3.7.1 Requirements. For a middleware that connects HPC resources and automated workflow orchestration, we identified the following requirements:

- open, well-documented standards-based APIs for remote access to HPC resources,
- secure, auditable authentication and authorization,
- support of complete job and data lifecycle management,
- stable service endpoints suitable for long-term operation,
- integration with institutional identity management systems and HPC schedulers,
- uniform access to heterogeneous HPC resources.

3.7.2 HPC Integration. To satisfy the HPC integration requirements outlined above, UNICORE [6, 64] is used as the middleware layer connecting Apache Airflow to external HPC resources. UNICORE is an open-source, state-of-the-art middleware stack that provides a uniform abstraction over heterogeneous HPC infrastructures and schedulers, enabling portable computation across a wide range of HPC systems.

To enable Airflow to interact with HPC resources via UNICORE, a dedicated Airflow executor, the UNICOREExecutor, is provided by the *airflow-unicore-integration* project [13]. The project is released under a permissive BSD-3-clause license via PyPI. The UNICOREExecutor comes with a set of operators built around a common base, the `UnicoreGenericOperator`, which implements core functionality for job submission, monitoring, status retrieval, and output collection. Specialized operators extend this base class and provide convenience presets for different submission modes, such as staging and executing user scripts or submitting batch job scripts. In this work, batch jobs are submitted using the `UnicoreBSSOperator`, which extends `UnicoreGenericOperator` and stages a BSS script for execution via UNICORE.

By integrating the UNICOREExecutor, the ESID pipeline can offload computationally or memory intensive tasks, such as large-scale simulations, to HPC resources. This allows the pipeline to execute workloads exceeding the resource limits of a Kubernetes-based deployment.

4 Results

4.1 A full software stack for epidemic disease management

The core result of this paper is the full stack of scientific software as visualized on a high-level in Fig. 12. It enables users of the ESID web front-end to interact and filter reported case and simulated data and to allow for swift exploration of "what-if" questions. In regular intervals, public data as well as queued HPC simulations are fed to the database via the ETL pipeline. To include local data, LHAs can upload individual data sets via the front-end which is then transferred to the ESID back-end and integrated into the database. A proof-of-concept for the integration of private or local data is given in Section 4.2. Aside from a predefined set of scenarios that cover common intervention settings such as 50 % remote work or mask wearing in public settings, individual scenarios from authenticated users can be stored in the ESID back-end. To ensure data privacy, only authenticated

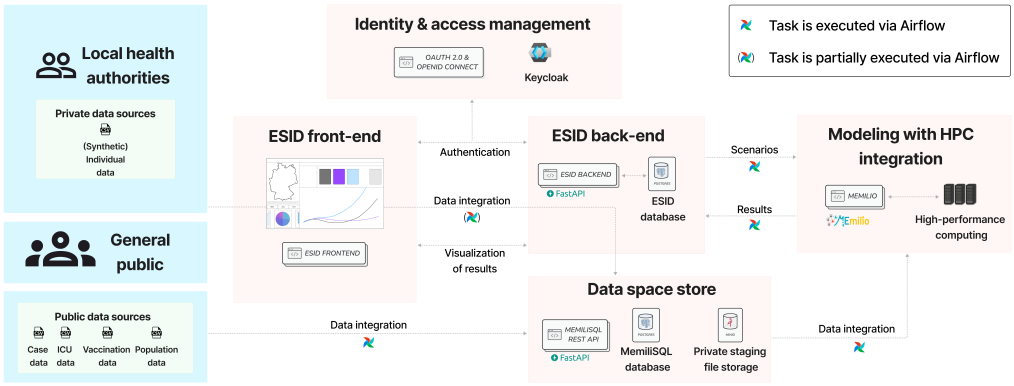

Fig. 12. **Overview of full the software stack.** On the left, we visualize public and potential LHA-specific data sets as well as users from the LHAs and the general public. Upon interaction with the ESID front-end and authentication, private data sources can be integrated. Furthermore, any interaction with the front-end triggers actions in the ESID back-end to retrieve data from the data store or to send scenario requests to the modeling and simulation stage with HPC.

users can upload data and view scenarios that were computed using individual data. In addition to scenarios with common interventions, an optimal control scenario is preconfigured. By default, the model is initialized with public data as in the pilot phase with only four LHAs, individual data is i) not available for the large majority of districts and ii) for any individual contribution, it is necessary to define if the data can be shared with other LHAs or districts. Upon updates of the integrated data, the scenarios are computed through MEmilio's metapopulation model, using the HPC resources to allow for a fast completion of tasks. The results are passed back to the ESID back-end and eventually visualized in the front-end. Whenever possible, tasks are executed automatically via Airflow to keep interactive latency low while maintaining traceability of every published scenario (inputs, seeds, image tags). For the open-source publication of the individual software components, see the section on software and data availability.

4.2 Proof-of-concept for LHA data integration

To test our platform's capability to integrate sensitive local data, we set up a proof-of-concept pipeline. In this pipeline, we generate synthetic local data through the MEmilio-ABM [34] offering individual information containing infection state and vaccination histories. With this local data set we are mimicking data that can be uploaded to our platform by LHAs. In this synthetic local data, we have, for any symptomatic, severe, and critical infection, direct information on the number of prior vaccinations which, in a real-world setting, could be obtained locally from infected individuals. However, on the country-wide scale, vaccination is reported separately from infections and intensive care treatments [38]. For the demonstration here, the synthetic data generated for the city of Cologne (see Fig. 13) is uploaded to our protected storage – which is not accessible by the broader public or other LHAs. In order to show the advantage of local information integration, we then initialize the population either with the full information ("full local data") or by assigning infections, with appropriate weights, randomly ("reduced information data") to the three immunity layers (naive, partial, and improved); see Fig. 14.

From Fig. 14, we see that the original, local data set contains 155 critical infections in the naive immunity layer at the beginning of the simulation while the randomized initialization only assigns

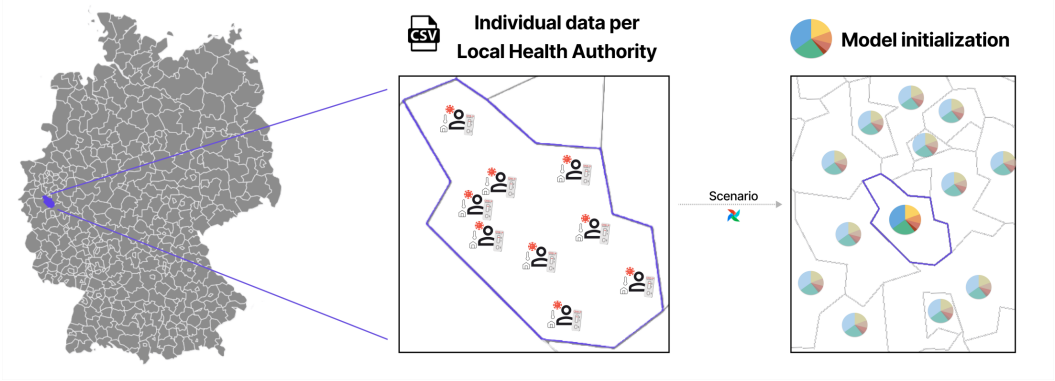

Fig. 13. **LHA data integration and use in Germany.** Available LHA data on individual level is used to improve accuracy and granularity in scenario simulations. Availability for data in the exemplary district of Cologne is shown in the left part of the figure. The improved initialization of a metapopulation model in this district is depicted by the brighter display of the population cohorts in the selected district in the right picture.

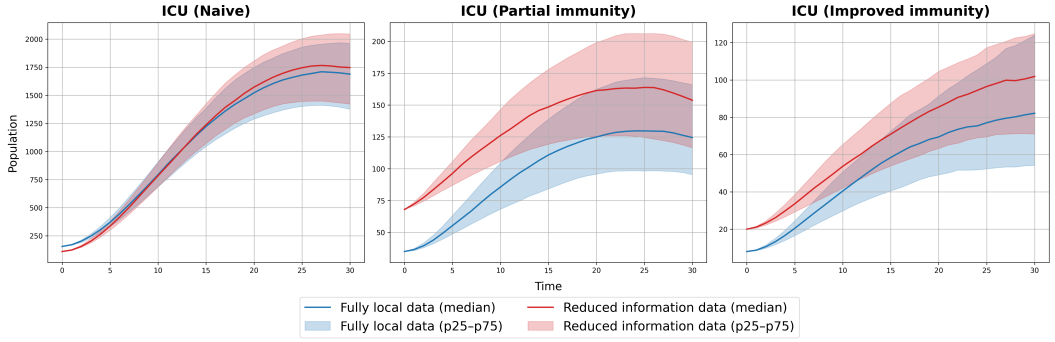

Fig. 14. **Outcomes of two simulations with synthetic local and global data.** We show simulations with a steep increase in ICU numbers over 30 days only differing in initial conditions. Initial infections are either assigned with full local information to immunity and vaccination layers (blue) or, where immunity layers and infection data are decoupled on the global scale, conducted based on appropriate random weights (red).

110 critical infections to this layer. Then, during the simulation, the situation turns around, with 1688 critical infections for the local data set and 1746 critical infections for the data set with reduced information in the naive immunity layer . In total, while both simulations start with 198 critical infections, at the end of the simulation horizon the result based on the original data set yields 1894 critical infections versus 2002 in the reduced data set, an increase by 5.7 %. While it has to be noted that the simulation represents a drastic increase of intensive care treatments in absolute numbers, the relative increase can be considered to be representative over many other scenarios and in cases where intensive care capacity is close to be exhausted a difference of 5-6 % in patient numbers can become considerable.

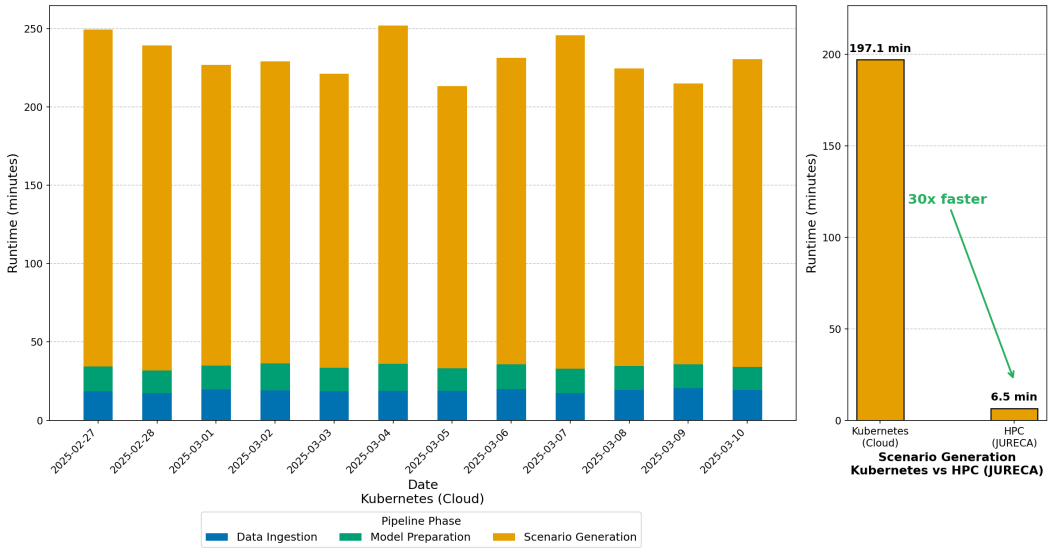

Fig. 15. **Daily end-to-end runtime distribution of the ESID pipeline measured over the evaluation window (27 February to 10 March 2025).** On the left, results for Kubernetes-based execution are shown as a stacked breakdown by pipeline phases. On the right, a direct comparison of the scenario generation phase under Kubernetes and when offloaded to HPC resources via UNICORE on JURECA is shown.

4.3 End-to-end runtime analysis

For a platform intended for epidemic or pandemic settings, performance is crucial. We therefore monitored the ESID pipeline as our core element for automated data ingestion, model preparation, scenario generation including simulation and data upload, in a preliminary version, over a representative evaluation period from 27 February to 10 March 2025. The goal was to analyze the runtime behavior of the complete workflow during regular operation. This evaluation targets the pipeline as an integrated system and does not aim to benchmark the underlying orchestration platform (i.e., Airflow running on Kubernetes). The analysis focuses on the stability and execution time of the necessary daily runs. In addition to the baseline Kubernetes-based execution, we analyzed the effect of offloading the computationally dominant task of the scenario generation phase to the JURECA HPC cluster.

Each task in the pipeline defines its CPU and memory requirements using Kubernetes resource specifications. Typically, tasks related to data ingestion are lightweight and request 1 vCPU and 4 GB of memory, with upper limits of 4 vCPUs and 12 GB of memory. Tasks in the compute-intensive phases, namely model preparation and scenario generation, are assigned more resources with requests of 2 vCPUs and 32 GB of memory, and limits up to 8 vCPUs and 64 GB of memory. For offloading to HPC resources, we use a single JURECA compute node consisting of two AMD EPYC processors (2×64 cores) and 512 GB of DDR4 memory.

Fig. 15 shows the daily end-to-end runtimes of the ESID pipeline over the evaluation period, presented as a stacked distribution across pipeline phases. Across the runs, the total execution time varies only slightly and stays within a narrow range, indicating stable behaviour at the DAG level. The runtime breakdown by phase shows that the scenario generation phase (represented by the `compute-scenarios` step) consumes most of the execution time, typically more than 80 % of the total runtime. Data ingestion and model preparation tasks contribute only a small and stable

fraction. The scenario generation phase is the most compute-intensive part of the pipeline, as it runs simulations for 33 scenarios on a coupled graph of 400 districts with ten runs each and subsequently uploads the results to the ESID back-end. When executed on a single HPC node, this phase achieves a 30-fold speedup, reducing the computation time from about 197 minutes to 6–7 minutes.

The evaluation shows that the pipeline can support workloads with varying requirements by leveraging both cloud and HPC resources, where the execution environment for each task is explicitly specified in the DAG definition.

4.4 Testing and user experience

While technical checks and the provision of source code are minimal requirements for our platform to be reusable, we continuously tested our platform with the four pilot health authorities, from a user's perspective, to ensure its usefulness in a novel epidemic setting. An inclusive development process with a regular user-feedback loop and a rigorous user testing strategy was employed to ensure that the developed platform meets the requirements of all stakeholders and end-users. To do this, product requirements and user stories were collected from LHAs and further refined throughout the entire development process.

As performance and immediate response are most critical for a useful web application, we minimized the size of the bundled scripts and assets. Libraries were carefully selected and optimized, with attention to dependency trees and tree-shaking, to keep the final bundle size as small as possible, currently resulting in a bundle size of just below 500 KB. Code splitting and lazy loading were employed to load certain functionalities only when needed. As not all functionality was required from the start, additional features are loaded on the fly. Using a consumer laptop, we measured the initial load time of the web front-end down for the first contentful paint as 0.8 seconds. Furthermore, we used Lighthouse [43], a tool to objectively test multiple characteristics of websites, to evaluate page load time, accessibility, and best practices along a continuous integration (CI) workflow, when new features were added, or when triggered manually. The Lighthouse performance score is overall 93 %. This metric is derived as a weighted score of several common page load metrics, such as the time for the first contentful paint and the total blocking time until user interaction is possible. In the Accessibility audit, which tracks accessibility issues based on WCAG guidelines [15] with the page, the web application scores 90 %. In the "Best Practices" section, our web application achieves 96 %.

In addition, we evaluated the platform through its front-end in the form of a workshop to assess two main objectives: 1) the intuitiveness and user-friendliness of the user interface and 2) the added value that the ESID front-end offers to LHAs. Item (1) focused on the first impression and intuitive usability of the platform. Here, we evaluated the coherence of the UI components (map, scenarios, diagrams) and various design elements. Item (2) aimed to conduct an overall assessment of the added value of the platform for various tasks in health authorities – such as monitoring, evaluation, and decision support – as well as the satisfaction of LHAs with the platform.

Through the qualitative evaluation of five data sets – as the workshop was conducted with a small group of LHA test users – a proof of concept was achieved. The results were collected from moderated feedback groups within the workshop and in the form of surveys on objectives (1) and (2). The collected ratings, which were based on multi-level scales, were used to create a series of selected key performance indicators (KPIs) – Net Promoter Score (NPS), Customer Satisfaction Score (CSS), and System Usability Scale (SUS).

First, we asked test users to rate on a scale of 1 to 10 how likely they would recommend the platform to another LHA. Individual ratings showed a rather neutral attitude towards the platform, resulting overall in a Net Promoter Score (NPS) of -20. The NPS has a range of -100 to +100 calculated

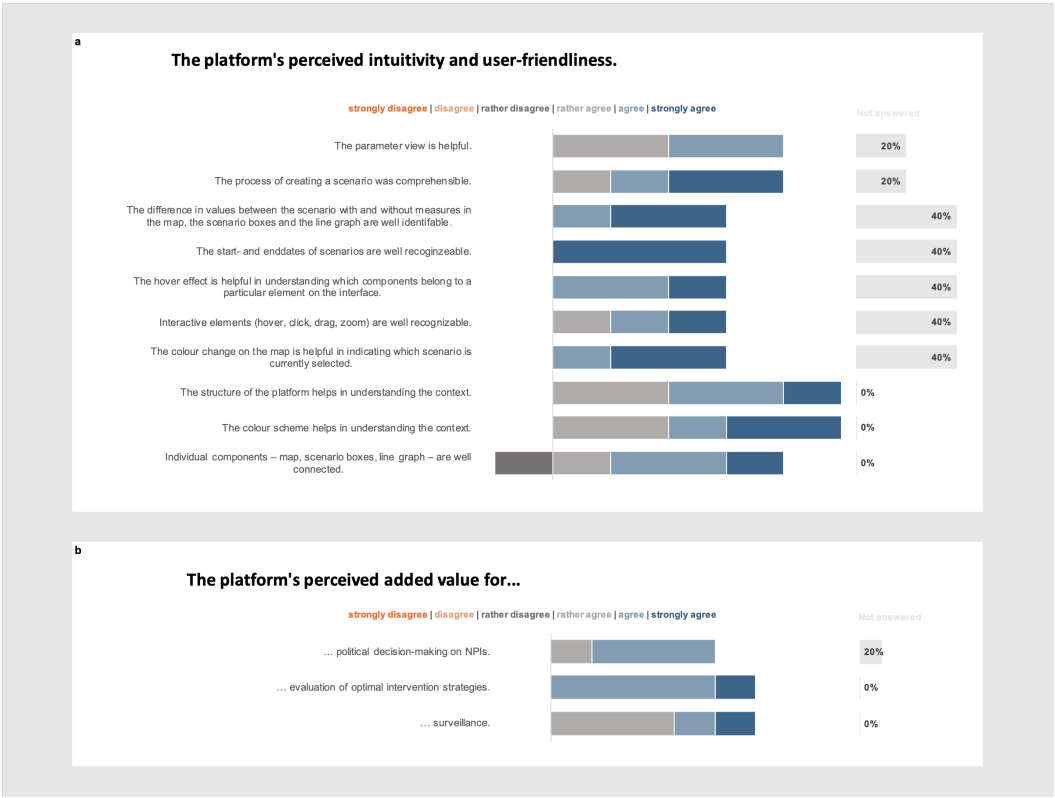

Fig. 16. **Evaluation on intuitiveness, user-friendliness, and added value of the platform.** This figure shows the responses of test users from different LHAs (n=5) to questions about the intuitiveness and user-friendliness of the platform (Panel a) and its perceived added value (Panel b). The survey questions have been transformed into statements with respective participant's ratings on a 6-point Likert scale ranging from strongly disagree (orange) to strongly agree (blue). The results are provided in percentage, visually represented by bars of different colors and separate bars for "not answered". The size of the bars corresponds to the number of responses, i.e., the relative percentage within a category.

by subtracting the percentage of "detractors" (scores 0-6) from the percentage of "promoters" (scores 9-10). A higher score indicates greater customer loyalty and a stronger potential for growth. As overall promotion without testing in a real pandemic setting might be difficult to judge, we then evaluated more specific aspects with more targeted questions; see Fig. 16.

Fig. 16a) shows an overall good reception for multiple questions – focusing on usability and intuitiveness such as the accessibility and connection of different components such as the maps, scenario boxes and the line graph, the platform's structure and color scheme, as well as interactive elements such as hover, click, drag and zoom. None of the test users had a negative impression or experienced difficulties understanding and using the platform. The user interface was rated as clearly structured and understandable. The relationship between the map view, scenario boxes and diagrams is comprehensible. Nearly 74 % of user responses were positive (agree or strongly agree). The overall satisfaction of test users with the platform was collected in a Customer satisfaction Score (CSS). The CSS scored 80 % satisfied test users. It was collected by asking users to rate their

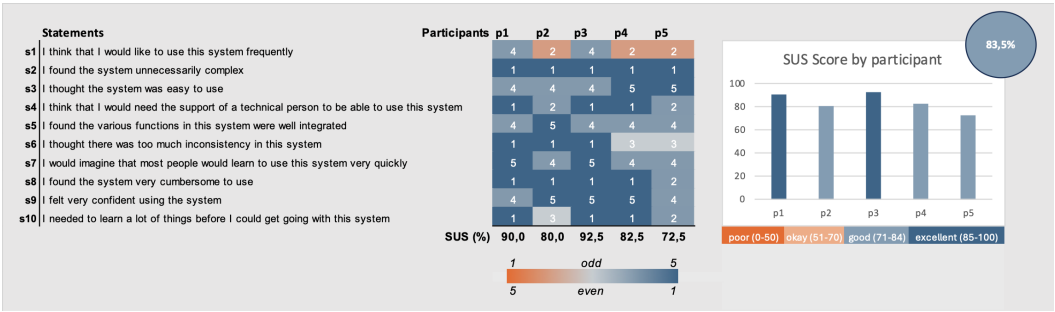

Fig. 17. **System Usability Scale (SUS) score.** This figure shows the 10 statements of a standardized SUS evaluation addressing a system's ease of use, complexity, consistency, and learnability. The structure follows an alternating pattern of positive and negative statements that users rated on a 5-point Likert scale (from "strongly disagree" to 'strongly agree"). The heat map visualizes the ratings of 5 test users each representing a different LHA (p1-p5). The numbers in the heat map are the actual ratings that were provided. In order to calculate the SUS score 1 needs to be subtracted from the user's Likert ratings for odd-numbered statements and the the user's Likert ratings for even-numbered statements have to be subtracted from 5. Each item score will range from 0 to 4. The sum of the numbers multiplied by 2.5 will result in a score between 0 and 100. The heatmap's color scheme reflects the alternating structure of the statements in line with the color scheme of the SUS score categories in the bar chart - from poor/dark orange to excellent/dark blue. Poor (0-50); okay (51-70); good (71-84); excellent (85-100). The bar chart depicts the individual SUS each test user scored, reaching an overall SUS score of 83.5 %.

experience on a numerical scale from 1-6. To calculate the percentage the number of satisfied customers, i.e., users, was divided by the total number of responses, then multiplied by 100.

In Fig. 16b), test users were asked to share their perception of the added value for routine tasks at LHAs, for which a continuous feedback loop throughout the project had identified an utility for the platform. The perceived added value of the platform in supporting LHAs in monitoring and early detection of outbreaks (surveillance) was fully recognized by one of the five test users. Another test user agreed that the platform may offer added value for surveillance, while three other rather agreed. In contrast, respondents are unanimously convinced that the platform offers significant advantages for assessing regional infection dynamics and evaluating the implementation of optimal intervention strategies, particularly the NPIs presented by the platform's scenarios. The function for displaying neighboring districts and the integration of local (regional) data were highlighted by LHAs as important features for regional risk management. Ultimately, the platform could be a useful tool for strengthening recommendations to decision makers with data-driven forecasts.

Finally, we applied the System usability Scale (SUS), a standardized method to measure the subjective usability of a system, such as the platform. A higher score indicates better perceived usability. The platform received a SUS score of 83.5 %, within the upper level of the "good" category and close to excellent, as visualized by Fig. 17. The platform is generally perceived as intuitive and clear, particularly in terms of usability and visual feedback. The test users unanimously stated that they feel confident using the platform and can operate it without extensive prior knowledge or technical guidance.

5 Conclusion

We presented a fully integrated software stack for epidemic and pandemic disease management with variable spatial resolutions, specific infection parameters and flexible intervention scenarios. The software components include data acquisition, transformation, ingestion to mathematical modeling

and visualization leveraging a maximum of automation and HPC integration. We provided end-to-end runtime profiling with more than 30 scenarios on a large-scale model, demonstrating that our pipeline is a) capable of being run overnight on a daily basis or b) with the use of HPC resources in just several minutes. To the best of our knowledge, no such software stack for infectious disease forecasting workflows exists. We showed that with a high level of automation, large modeling tasks can be effectively accelerated to a level that makes it applicable in real world settings while avoiding manual interaction and efficiently using HPC resources.

A corner stone of our project was the testing and evaluation of the user experience with our cohort of pilot LHAs and their employees to ensure a development that fits the users' needs in pandemic control and mitigation. We, thus, investigated specific requirements such as the intuitive use to avoid substantial instruction delays or visualization and filtering along the multiple dimensions of infectious disease. 74 % of user responses agreed or strongly agreed on user-friendliness and added value.

While we aimed at supporting public health experts and decision makers with mathematical modeling, our results satisfy requirements of a surveillance system. For the latter, a combination with early warning systems and indicators through wastewater monitoring would be beneficial to support outbreak detection. Then, daily automated pipelines could provide a well structured and spatially resolved view on ongoing disease dynamics and outbreaks.

The core components for mathematical modeling and visual analytics with the connecting REST API were designed and developed with a maximum of generality to allow for model adaptations to different spatial resolutions or courses of the disease. Thus, our publicly available open-source components can easily be adapted to different (administrative) units within the same country or transferred to other countries as long as corresponding data is available. More importantly, in the context of pandemic preparedness, the computational framework can be applied to other respiratory pathogens with pandemic potential. This requires an adaptation of the disease model and front-end visualization by redefinition of disease parameters, groups and compartments, which is feasible because of the modular architecture of the framework.

6 Acknowledgement

The authors gratefully acknowledge computing time on the supercomputer JURECA [32] at Forschungszentrum Jülich under grant no. `HPC4EPI`. The authors furthermore gratefully acknowledge the support of and exchange with Torge Korff, Annelene Kossow, Thomas Dau, Thies Marquardt, Susanne Krause from LHAs and Alexander Ullrich from the RKI. The authors thank Uwe Naumann (RWTH Aachen University, i12 Software and Tools for Computational Engineering) for providing code for automatic differentiation. This work was supported by the Initiative and Networking Fund of the Helmholtz Association (grant agreement number KA1-Co-08, Project LOKI-Pandemics) and partially funded by the German Federal Ministry for Digital and Transport under grant agreement FKZ19F2211A and FKZ19F2211B (Project PANDEMOS).

Software and data availability

The mathematical models and scenario configurations are implemented in the MEmilio framework [9]. The specific code is openly available at https://github.com/SciCompMod/memilio/tree/esid-scenarios. The dagbags without secrets are available at https://codebase.helmholtz.cloud/loki/public-dagbag. The code for airflow to call is https://codebase.helmholtz.cloud/loki/memiliflow. The public docker image with our packages installed can be found at registry.hzdr.de/loki/memiliflow:main. The ESID front-end is available at https://github.com/DLR-SC/ESID and the back-end at https://github.com/SciCompMod/ESID-backend. The Fleet/K8s deployment code is openly available

at https://codebase.helmholtz.cloud/loki/public-fleet-deployment-memilisql-restapi. The memilisql/data integration API code is openly available at https://codebase.helmholtz.cloud/loki/public-memilisql-restapi.

References

[1] Dan Abramov. 2025. *Redux.* https://redux.js.org/

[2] Dan Abramov. 2025. *Redux Toolkit.* https://redux-toolkit.js.org/tutorials/rtk-query/

[3] Oriol Abril-Pla, Valerio Andreani, Colin Carroll, Leo Dong, Christopher J. Fonnesbeck, Maxim Kochurov, Ravin Kumar, Jim Lao, Christian C. Luhmann, Osvaldo A. Martin, Matthias Osthege, Ricardo Vieira, Thomas Wiecki, and Rob Zinkov. 2023. PyMC: a modern, and comprehensive probabilistic programming framework in Python. *PeerJ Computer Science* 9 (2023), e1516. https://doi.org/10.7717/peerj-cs.1516

[4] R. Adhikari, Austen Bolitho, Fernando Caballero, Michael E. Cates, Jakub Dolezal, Timothy Ekeh, et al. 2020. Inference, prediction and optimization of non-pharmaceutical interventions using compartment models: the PyRoss library. arXiv:2005.09625 https://arxiv.org/abs/2005.09625

[5] Michelle Barker, Neil P. Chue Hong, Daniel S. Katz, Anna-Lena Lamprecht, Carlos Martinez-Ortiz, Fotis Psomopoulos, Jennifer Harrow, Leyla Jael Castro, Morane Gruenpeter, Paula Andrea Martinez, and Tom Honeyman. 2022. Introducing the FAIR Principles for research software. *Scientific Data* 9, 1 (2022), 622. https://doi.org/10.1038/s41597-022-01710-x

[6] K. Benedyczak, B. Schuller, Maria Petrova-El Sayed, J. Rybicki, and Richard Grunzke. 2016. UNICORE 7 — Middleware services for distributed and federated computing. *2016 International Conference on High Performance Computing & Simulation (HPCS)* (2016), 613–620. https://doi.org/10.1109/HPCSim.2016.7568392

[7] John T Betts. 2010. *Practical methods for optimal control and estimation using nonlinear programming.* SIAM.

[8] Pawandeep Kaur Betz, Julien Stoll, Valerie Grappendorf, Jonas Gilg, Moritz Zeumer, et al. 2023. ESID: Exploring the Design and Development of a Visual Analytics Tool for Epidemiological Emergencies. In *2023 IEEE VIS Workshop on Visualization for Pandemic and Emergency Responses (Vis4PandEmRes).* 8–14. https://doi.org/10.1109/Vis4PandEmRes60343.2023.00007

[9] Julia Bicker, Carlotta Gerstein, David Kerkmann, Sascha Korf, René Schmieding, Anna Wendler, Henrik Zunker, Daniel Abele, Maximilian Betz, Khoa Nguyen, Lena Plötzke, Kilian Volmer, Agatha Schmidt, Nils Waßmuth, Patrick Lenz, Daniel Richter, Hannah Tritzschak, Ralf Hannemann-Tamas, Julian Litz, Paul Johannssen, Marielena Borges, Annika Jungklaus, Manuel Heger, Annalena Lange, Elisabeth Kluth, Kathrin Rack, Vincent Wieland, Jonas Arruda, Sebastian Binder, Margrit Klitz, Martin Siggel, Manuel Dahmen, Achim Basermann, Michael Meyer-Hermann, Jan Hasenauer, and Martin J. Kühn. 2026. MEmilio – A high performance Modular EpideMIcs simuLatIOn software for multi-scale and comparative simulations of infectious disease dynamics. (2026). https://doi.org/10.48550/arXiv.2602.11381

[10] Julia Bicker, David Kerkmann, Sascha Alexander Korf, Lena Plötzke, René Schmieding, Anna Clara Wendler, Henrik Zunker, Khoa Nguyen, Daniel Abele, Carlotta Gerstein, Patrick Lenz, Maximilian Franz Betz, Agatha Schmidt, Ralf Hannemann-Tamas, Kilian Volmer, Nils Waßmuth, Hannah Tritzschak, Daniel Richter, Manuel Heger, Achim Basermann, Michael Meyer-Hermann, Jan Hasenauer, and Martin Joachim Kühn. [n. d.]. *MEmilio v2.1.0 - A high performance Modular EpideMIcs simuLatIOn software.* https://elib.dlr.de/213614/

[11] Julia Bicker, René Schmieding, Michael Meyer-Hermann, and Martin J. Kühn. 2025. Hybrid metapopulation agent-based epidemiological models for efficient insight on the individual scale: A contribution to green computing. *Infectious Disease Modelling* 10, 2 (2025), 571–590. https://doi.org/10.1016/j.idm.2024.12.015

[12] BMAS. 2020. Pendlerverflechtungen der sozialversicherungspflichtig Beschäftigten nach Kreisen - Deutschland (Jahreszahlen). https://statistik.arbeitsagentur.de/DE/Navigation/Statistiken/Interaktive-Statistiken/Pendler/Pendler-Nav.html https://statistik.arbeitsagentur.de/DE/Navigation/Statistiken/Interaktive-Statistiken/Pendler/Pendler-Nav.html.

[13] Christian Böttcher. 2025. airflow-unicore-integration: Airflow provider for UNICORE interactions. https://pypi.org/project/airflow-unicore-integration/. Version 0.1.4, accessed 2025-11-25.

[14] Bundesministerium für Gesundheit. [n. d.]. Digitales Gesundheitsamt: IT-Plattform ÖGD. https://gesundheitsamt-2025.de/digitalisierung/it-plattform-oegd#top

[15] World Wide Web Consortium. 2026. *WCAG 2.* https://www.w3.org/WAI/standards-guidelines/wcag/

[16] Jonas Dehning, Sebastian B. Mohr, Sebastian Contreras, Philipp Dönges, Emil N. Iftekhar, Oliver Schulz, Philip Bechtle, and Viola Priesemann. 2023. Impact of the Euro 2020 championship on the spread of COVID-19. *Nature Communications* 14, 1 (2023). https://doi.org/10.1038/s41467-022-35512-x

[17] Angel N. Desai, Moritz U. G. Kraemer, Sangeeta Bhatia, Anne Cori, Pierre Nouvellet, Mark Herringer, Emily L. Cohn, Malwina Carrion, John S. Brownstein, Lawrence C. Madoff, and Britta Lassmann. 2019. Real-time Epidemic Forecasting: Challenges and Opportunities. *Health Security* 17, 4 (2019), 268–275. https://doi.org/10.1089/hs.2019.0022

[18] Statistisches Bundesamt (DESTATIS). [n. d.]. Population: Germany, reference date, age. https://www-genesis.destatis.de/datenbank/online/statistic/12411/table/12411-0005

[19] Ensheng Dong, Hongru Du, and Lauren Gardner. 2020. An interactive web-based dashboard to track COVID-19 in real time. *The Lancet Infectious Diseases* 20, 5 (2020), 533–534. https://doi.org/10.1016/S1473-3099(20)30120-1

[20] European Centre for Disease Prevention and Control. 2024. *Public health and social measures for health emergencies and pandemics in the EU/EEA: recommendations for strengthening preparedness planning.* Technical Report. https://doi.org/10.2900/253991

[21] Luca Ferretti, Chris Wymant, Michelle Kendall, Lele Zhao, Anel Nurtay, Lucie Abeler-Dörner, Michael Parker, David Bonsall, and Christophe Fraser. 2020. Quantifying SARS-CoV-2 transmission suggests epidemic control with digital contact tracing. *science* 368, 6491 (2020), eabb6936.

[22] Elena Fischer, K. Stein, Torben Heinsohn, Martin Kühn, Berit Lange, and Carolina J. Klett-Tammen. 2025. Global efforts to develop and integrate infectious disease forecasting platforms for public health decision-making: a scoping review protocol. (2025). https://osf.io/q32kg In preparation..

[23] Federal Ministry for Digital Transformation and Government Modernisation. 2025. *Bund ID.* https://id.bund.de/en/

[24] Cloud Native Computing Foundation. 2025. *KeyCloak - Open Source Identity and Access Management.* https://www.keycloak.org/

[25] Laura Fumanelli, Marco Ajelli, Piero Manfredi, Alessandro Vespignani, and Stefano Merler. 2012. Inferring the Structure of Social Contacts from Demographic Data in the Analysis of Infectious Diseases Spread. *PLoS Computational Biology* 8, 9 (2012), e1002673. https://doi.org/10.1371/journal.pcbi.1002673

[26] Sebastian Funk, Anton Camacho, Adam J. Kucharski, Rosalind M. Eggo, and W. John Edmunds. 2018. Real-time forecasting of infectious disease dynamics with a stochastic semi-mechanistic model. *Epidemics* 22 (2018), 56–61. https://doi.org/10.1016/j.epidem.2016.11.003

[27] The PostgreSQL Global Development Group. 2025. *PostgreSQL.* https://www.postgresql.org/

[28] Dick Hardt. 2012. The OAuth 2.0 Authorization Framework. RFC 6749. https://doi.org/10.17487/RFC6749

[29] Joe Hasell, Edouard Mathieu, Diana Beltekian, Bobbie Macdonald, Charlie Giattino, Esteban Ortiz-Ospina, Max Roser, and Hannah Ritchie. 2020. A cross-country database of COVID-19 testing. *Scientific Data* 7, 1 (2020), 345. https://doi.org/10.1038/s41597-020-00688-8

[30] Robert Hinch, William J. M. Probert, Anel Nurtay, Michelle Kendall, Chris Wymant, Matthew Hall, et al. 2021. OpenABM-Covid19—An agent-based model for non-pharmaceutical interventions against COVID-19 including contact tracing. *PLOS Computational Biology* 17, 7 (2021), 1–26. https://doi.org/10.1371/journal.pcbi.1009146

[31] John (Xuefeng) Jiang, Peter Cram, Kangkang Qi, and Ge Bai. 2024. Challenges and dynamics of public health reporting and data exchange during COVID-19: insights from US hospitals. *Health Affairs Scholar* 2, 1 (2024), qxad080. https://doi.org/10.1093/haschl/qxad080

[32] Jülich Supercomputing Centre. 2021. JURECA: Data Centric and Booster Modules implementing the Modular Supercomputing Architecture at Jülich Supercomputing Centre. *Journal of large-scale research facilities* 7, A182 (2021). https://doi.org/10.17815/jlsrf-7-182

[33] D.A. Keim. 2002. Information visualization and visual data mining. *IEEE Transactions on Visualization and Computer Graphics* 8, 1 (2002), 1–8. https://doi.org/10.1109/2945.981847

[34] David Kerkmann, Sascha Korf, Khoa Nguyen, Daniel Abele, Alain Schengen, Carlotta Gerstein, Jens Henrik Göbbert, Achim Basermann, Martin J. Kühn, and Michael Meyer-Hermann. 2025. Agent-based modeling for realistic reproduction of human mobility and contact behavior to evaluate test and isolation strategies in epidemic infectious disease spread. *Computers in Biology and Medicine* 193 (2025), 110269. https://doi.org/10.1016/j.compbiomed.2025.110269

[35] Cliff C. Kerr, Robyn M. Stuart, Dina Mistry, Romesh G. Abeysuriya, Katherine Rosenfeld, Gregory R. Hart, et al. 2021. Covasim: An agent-based model of COVID-19 dynamics and interventions. *PLOS Computational Biology* 17, 7 (2021), 1–32. https://doi.org/10.1371/journal.pcbi.1009149

[36] Robert Koch-Institut. [n. d.]. SurvNet database. https://survnet.rki.de/Default.aspx

[37] Robert Koch-Institut. 2024. *COVID-19-Impfungen in Deutschland.* https://doi.org/10.5281/zenodo.12697471

[38] Robert Koch-Institut. 2025. *Intensivkapazitäten und COVID-19-Intensivbettenbelegung in Deutschland.* https://doi.org/10.5281/zenodo.17623109

[39] Robert Koch-Institut. 2025. *SARS-CoV-2 Infektionen in Deutschland.* https://doi.org/10.5281/zenodo.17627234

[40] Wadim Koslow, Martin J. Kühn, Sebastian Binder, Margrit Klitz, Daniel Abele, Achim Basermann, and Michael Meyer-Hermann. 2022. Appropriate relaxation of non-pharmaceutical interventions minimizes the risk of a resurgence in SARS-CoV-2 infections in spite of the Delta variant. *PLOS Computational Biology* 18, 5 (2022), e1010054. https://doi.org/10.1371/journal.pcbi.1010054

[41] Martin J. Kühn, Daniel Abele, Tanmay Mitra, Wadim Koslow, Majid Abedi, Kathrin Rack, et al. 2021. Assessment of effective mitigation and prediction of the spread of SARS-CoV-2 in Germany using demographic information and spatial resolution. *Mathematical Biosciences* (2021), 108648. https://doi.org/10.1016/j.mbs.2021.108648

[42] Berit Lange, Veronika Jäger, Viktoria Rücker, Max Hassenstein, Manuela Harries, Reinhard Berner, et al. 2022. *Interimsanalyse des IMMUNEBRIDGE-Projektes zur Kommunikation von vorläufigen Ergebnissen an die Modellierungskonsortien der BMBF-geförderten Modellierungsplattform.* https://doi.org/10.5281/zenodo.6968574
[43] Google LLC. 2025. *Lighthouse.* https://developer.chrome.com/docs/lighthouse/
[44] Google LLC. 2025. *Material Design.* https://m3.material.io/
[45] Redis Ltd. 2025. *Redis.* https://redis.io/
[46] Shahbaz Memon, Johann F. Jadebeck, Michael Osthege, Anna Wendler, David Kerkmann, Henrik Zunker, Wolfgang Wiechert, Katharina Nöh, Jens Henrik Göbbert, Björn Hagemeier, Morris Riedel, and Martin J. Kühn. 2024. Automated Processing of Pipelines Managing Now- and Forecasting of Infectious Diseases. In *2024 47th MIPRO ICT and Electronics Convention (MIPRO).* 1157–1162. https://doi.org/10.1109/MIPRO60963.2024.10569336
[47] José Sebastián Ramírez Montaño. 2025. *FastAPI.* https://fastapi.tiangolo.com/
[48] Joél Mossong, Niel Hens, Mark Jit, Philippe Beutels, Kari Auranen, Rafael Mikolajczyk, et al. 2008. Social Contacts and Mixing Patterns Relevant to the Spread of Infectious Diseases. *PLOS Medicine* 5, 3 (2008), e74. https://doi.org/10.1371/journal.pmed.0050074
[49] Sebastian A. Müller, Michael Balmer, William Charlton, Ricardo Ewert, Andreas Neumann, Christian Rakow, Tilmann Schlenther, and Kai Nagel. 2021. Predicting the effects of COVID-19 related interventions in urban settings by combining activity-based modelling, agent-based simulation, and mobile phone data. *PLOS ONE* 16, 10 (2021), 1–32. https://doi.org/10.1371/journal.pone.0259037
[50] Uwe Naumann. 2011. *The art of differentiating computer programs: an introduction to algorithmic differentiation.* SIAM.
[51] Jasmina Panovska-Griffiths, Cliff C Kerr, Robyn M Stuart, Dina Mistry, Daniel J Klein, et al. 2020. Determining the optimal strategy for reopening schools, the impact of test and trace interventions, and the risk of occurrence of a second COVID-19 epidemic wave in the UK: a modelling study. *The Lancet Child & Adolescent Health* 4, 11 (2020), 817–827. https://doi.org/10.1016/S2352-4642(20)30250-9
[52] Meta Platforms. 2025. *React.* https://react.dev/
[53] Lena Plötzke, Anna Wendler, René Schmieding, and Martin J. Kühn. 2026. Revisiting the Linear Chain Trick in epidemiological models: Implications of underlying assumptions for numerical solutions. *Mathematics and Computers in Simulation* 239 (2026), 823–844. https://doi.org/10.1016/j.matcom.2025.07.045
[54] Kiesha Prem, Alex R. Cook, and Mark Jit. 2017. Projecting social contact matrices in 152 countries using contact surveys and demographic data. *PLoS computational biology* 13, 9 (2017), e1005697. https://doi.org/10.1371/journal.pcbi.1005697
[55] Akond Rahman, Rezvan Mahdavi-Hezaveh, and Laurie Williams. 2019. A systematic mapping study of infrastructure as code research. *Information and Software Technology* 108 (2019), 65–77. https://doi.org/10.1016/j.infsof.2018.12.004
[56] Robert C. Reiner, Ryan M. Barber, James K. Collins, Peng Zheng, Christopher Adolph, James Albright, et al. 2021. Modeling COVID-19 scenarios for the United States. *Nature Medicine* 27, 1 (2021), 94–105. https://doi.org/10.1038/s41591-020-1132-9
[57] Nat Sakimura, John Bradley, and Naveen Agarwal. 2015. Proof Key for Code Exchange by OAuth Public Clients. RFC 7636. https://doi.org/10.17487/RFC7636
[58] N. Sakimura, J. Bradley, M. Jones, B. de Medeiros, and C. Mortimore. 2023. *OpenID Connect Core 1.0.* https://openid.net/specs/openid-connect-core-1_0.html
[59] Material UI SAS. 2025. *MUI.* https://mui.com/
[60] Susanne Sass and Alexander Mitsos. 2025. Obscured by terminology: Hidden parallels in direct methods for open-loop optimal control. *Journal of Process Control* 154 (2025), 103513.
[61] Agatha Schmidt, Henrik Zunker, Alexander Heinlein, and Martin J. Kühn. 2026. Graph neural network surrogates to leverage mechanistic expert knowledge towards reliable and immediate pandemic response. *Scientific Reports* 16, 1 (Feb. 2026). https://doi.org/10.1038/s41598-026-39431-5
[62] Andrew J. Shattock, Epke A. Le Rutte, Robert P. Dünner, Swapnoleena Sen, Sherrie L. Kelly, Nakul Chitnis, and Melissa A. Penny. 2022. Impact of vaccination and non-pharmaceutical interventions on SARS-CoV-2 dynamics in Switzerland. *Epidemics* 38 (2022), 100535. https://doi.org/10.1016/j.epidem.2021.100535
[63] Ask Solem. 2025. *Celery.* https://docs.celeryq.dev/en/stable/index.html/
[64] A. Streit, D. Erwin, Th. Lippert, D. Mallmann, R. Menday, M. Rambadt, M. Riedel, M. Romberg, B. Schuller, and Ph. Wieder. 2005. Unicore — From project results to production grids. In *Grid Computing The New Frontier of High Performance Computing*, Lucio Grandinetti (Ed.). Advances in Parallel Computing, Vol. 14. North-Holland, 357–376. https://doi.org/10.1016/S0927-5452(05)80018-8
[65] C. J. F. ter Braak and J. A. Vrugt. 2008. Differential Evolution Markov Chain with snooker updater and fewer chains. *Statistics and Computing* 18, 4 (2008), 435–446. https://doi.org/10.1007/s11222-008-9104-9
[66] Aki Vehtari, Andrew Gelman, Daniel Simpson, Bob Carpenter, and Paul-Christian Bürkner. 2021. Rank-Normalization, Folding, and Localization: An Improved $\widehat{R}$ for Assessing Convergence of MCMC (with Discussion). *Bayesian Analysis* 16, 2 (2021), 667 – 718. https://doi.org/10.1214/20-BA1221

[67] Ben Vermeulen, Matthias Müller, and Andreas Pyka. 2021. Social Network Metric-Based Interventions? Experiments with an Agent-Based Model of the COVID-19 Pandemic in a Metropolitan Region. *Journal of Artificial Societies and Social Simulation* 24, 3 (2021), 6. https://doi.org/10.18564/jasss.4571

[68] Andreas Wächter and Lorenz T Biegler. 2006. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming* 106, 1 (2006), 25–57.

[69] Anna Wendler, Lena Plötzke, Hannah Tritzschak, and Martin J. Kühn. 2026. A nonstandard numerical scheme for a novel SECIR integro-differential equation-based model allowing nonexponentially distributed stay times. *Appl. Math. Comput.* 509 (2026), 129636. https://doi.org/10.1016/j.amc.2025.129636

[70] Mark D Wilkinson et al. 2016. The FAIR Guiding Principles for scientific data management and stewardship. *Scientific Data 3, 160018* (2016).

[71] Christian Winkler and Tudor Mocanu. 2020. Impact of political measures on passenger and freight transport demand in Germany. *Transportation Research Part D: Transport and Environment* 87 (2020), 102476. https://doi.org/10.1016/j.trd.2020.102476

[72] World Health Organization. [n. d.]. WHO Coronavirus (COVID 19) Dashboard. https://covid19.who.int

[73] Moritz Zeumer, Jonas Gilg, Pawandeep Kaur Betz, and Andreas Gerndt. 2023. VVAFER — Versatile visual analytics framework for exploration and research. *Electronic Imaging* 35, 1 (2023), 402–1–402–1. https://doi.org/10.2352/EI.2023.35.1.VDA-402

[74] Henrik Zunker, Philipp Dönges, Patrick Lenz, Seba Contreras, and Martin J. Kühn. 2025. Risk-mediated dynamic regulation of effective contacts de-synchronizes outbreaks in metapopulation epidemic models. *Chaos, Solitons & Fractals* 199 (2025), 116782. https://doi.org/10.1016/j.chaos.2025.116782

[75] Henrik Zunker, René Schmieding, David Kerkmann, Alain Schengen, Sophie Diexer, Rafael Mikolajczyk, Michael Meyer-Hermann, and Martin J. Kühn. 2024. Novel travel time aware metapopulation models and multi-layer waning immunity for late-phase epidemic and endemic scenarios. *PLOS Computational Biology* 20, 12 (2024). https://doi.org/10.1371/journal.pcbi.1012630