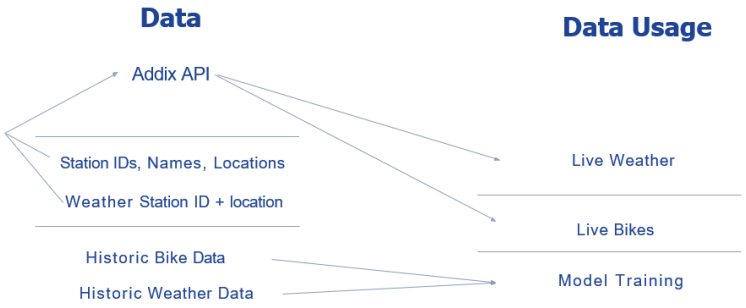
D4A Solution, Smart Bike System

Name of your institution and contact	KielRegion GmbH, l.kroeger@kielregion.de
Place of the implemented solution	KielRegion
Implementation level (e.g. internal, municipal, intercommunal, regional)	Regional
Dimension	• Technical • Organisational • Regulatory
Subdimension	• Interoperability • Data management • Data values
Problem	The SprottenFlotte bikesharing system in the KielRegion faces persistent issues with uneven bike distribution. Some stations are empty for long periods, while others remain overcrowded with unused bikes. The current redistribution process is based on internal models by the operator Donkey Republic but does not sufficiently reflect local needs or provide transparency to public stakeholders. Local authorities and users regularly report poor availability, reducing the system's attractiveness.
Solution	The aim is to solve the problem of unevenly distributed bikes in the KielRegion to avoid empty stations and unused bikes. Therefore a data-driven prediction tool based on historical usage data, bike availability and weather information was developed in cooperation with students from Kiel University of Applied Sciences. This will significantly improve the availability of the SprottenFlotte fleet, make it easier for users to find a bike and increase the overall utilisation of the system. The web-based tool, built with Streamlit, visualizes forecasted availability and helps service teams prioritize redistribution tasks more effectively. The data is updated hourly, with predictions covering the next 5 hours. All relevant station and weather data is managed through structured CSV files. A professional IT provider is needed for stable operation and data integration. Our solution is intended to be permanently integrated into the SprottenFlotte's operations. There is potential for scaling up to other regions, forms of mobility or integration into municipal mobility platforms.
Feedback	A key lesson learned is that prediction quality strongly depends on the availability and diversity of the input data. It also became clear that the project faced limitations in terms of data availability and interoperability, as it was based on too few different data sources and therefore did not meet the requirements for data interoperability.
Format (e.g. Open Source)	Publicly accessible via GitHub, but not licensed as Open Source.
Costs	The development was carried out as part of a university application project at the University of Applied Sciences Kiel and therefore incurred no direct costs. Implementation and operation rely on voluntary collaboration and existing infrastructure, so operational costs are currently negligible. While development was done in an academic setting at minimal cost, professional operation will require financial resources for IT services, including hosting, maintenance, and support.

Links	https://streamlit.io/ https://github.com/
Screenshots, visuals, images	

Architecture

The architecture of the Sprottenflotte Prediction Tool is centered around several interconnected Python modules, with the main application script `app.py` orchestrating the overall operation of the Streamlit web application. `app.py` serves as the core of the tool, initializing the Streamlit interface, managing user interactions, and setting up the page configuration. It handles model selection between Random Forest and Deep Learning, loads necessary data, and executes the prediction process. Supporting this, `app_functions.py` contains utility functions essential for data manipulation and interface operations, such as displaying messages, filtering data frames, and organizing station data for visualization.

Data management is handled by `data.py`, which retrieves and synchronizes data from external APIs and updates datasets on GitHub, ensuring that the application always uses the most recent data. Predictions are generated by the `predictions_dl.py` and `predictions_rf.py` modules, which process data through a BiLSTM model and a Random Forest model, respectively. These modules perform necessary data transformations and update the results for integration back into the main application. Together, these components form a cohesive infrastructure that supports real-time predictions and dynamic data visualization, ensuring the application is both scalable and maintainable.