Documentation

Here we document the currently implemented features of stratus.

This is a continuous work in progress. You are very welcome to detail and extend entries.

Standard functionality

Shared functionality is used by many widgets; providing a consistent and powerful experience out of the box.

Functionality	Features
`Space`	Hosts one or many `Widgets`. Can spawn, minimize or close widgets. Has a header with space-specific actions.
`Widget` Container of functionality with a clear and single purpose.	Modular approach which gives great flexibility to use-cases (compose widgets as needed) and keeps development scalable (limited complexity to each widget). Widgets are: • part of a context (i.e. projects, objects) • in a mode (i.e. Edit, View) • in a view (i.e. List, Diagram) Additionally, widgets can be: • loading: Content is not rendered, instead a loading animation is shown • in help mode: All help fields are expanded Each widget has: • context: Show what the root entity of the widget is, and/or number of filtered values • key figures: Show most important summary of the displayed content • header actions: Perform an action over the displayed content (i.e. export, enter/leave edit mode, ….) • alerts: Warn users if some displayed information might be out of date or incomplete. • data selection mode: Select what content is displayed (i.e. time range); using the `Filter` and `Parameter`. Selection criteria is shown on hover if outside the data selection mode. • no data dialog: Shown if data is not available, to help the user get started.
`DataSelection` Select data to display.	Provides input fields specific to the widget which influences what data is displayed. Most data is selected when the widget is created (i.e. the object master data widget is created for a specific object). Some widgets further are able to react to changes without recreating it, i.e. sidebars showing the currently selected object. The `DataSelection` contains the `Filter` and the `Parameters`.
`Filter` Filter the displayed data.	Select operators (equals, smaller, …) depending on the data type of the field. Known data types are `boolean`, `string`, `number` and `collections`. For fields which can be empty, show additional operators (`isSet`, `isNotSet`). For “negative” operators (such as `not in collection`), include empty fields in the resulting set. Enter the operator value in a control specific to the operator and data type (e.g. enter monetary values in `kCHF`). Chain operators using AND or OR. Operators then decide based on the raw value (not rounded) whether a row fulfills the condition or not. Hover over the filter icon in the widget header to see the currently applied filter.
`Parameter` Influence calculation.	Select parameters depending on the data shown in the widget. Parameters may have a predefined value (called a `Setting`) from the backend. Parameters can be shown in the simulation section. The simulation section provides sliders (linear, or arbitrary steps) directly in the widget (outside the `DataSelection` mode) so effects of mutated parameters are directly visible.
`Sort` Sort the displayed data.	Sort ascending or descending.
`KeyFigures` Summarize displayed content.	Show (description, value) pairs prominently to the user.
`HeaderActions` User-Interactions with the content.	Show primary actions directly in the widget header. Show secondary actions in the three-dot menu. Disable loading actions while their action is loading. Show confirmation menu for critical actions (i.e. delete).
`ExcelExport` Export as xlsx.	Export raw values (not rounded) of useful fields. Set width to expected content width. Optionally export formatted value (i.e. Usage as text).
`DataGrid` Compare and find entries.	Show columns of formatted useful fields. Set width of columns to the expected content width. Deactivate Columns and have this persisted over reloads. Sort undefined fields always last. Display visualizations of field values (i.e. colored dot for condition). Quicksearch over displayed values. Customized version of React Data Grid component - MUI X. Tightly integrated with the `Filter` and the `ExcelExport`.
`Offline Mode` & `Synchronization` Use and edit while offline.	Store changes while offline, which are processed as soon as online again. Keep failed requests (network, other reasons) in a queue. Show the user pending changes, and give an option to discard. Ask user when an edit would override edits previously stored to the server.

	Supported fully for `object`, `assessment`, `object history`, `project`, `project planning`, `object groups`. Partially supported in `constructional components`, `note`. Not supported for `energy mix`, `energy accounting`, `file`, `user` and others.
`Form` Utilities built for form fields	`FormErrors` shows a summary of the errors occurred in the form. `FormFieldStatusIndicator` shows status (error, warning, …) of form fields.
`Dialog`, `Tooltip`, `BarChart`, …	Often used components are wrapped to apply stratus-specific styling and behavior.
`Responsiveness`	Stratus works on different screen sizes, including smaller than 1080p.

Technical Foundation

Technical structures which power the more advanced features. Not complete; only includes the often used structures. More technical details can be found here.

Functionality	Features
`versionedEntityAdapter`	Store different versions of the same entity. Simple interface to query and update such entities.
`synchronizationSlice`	Keep queue of pending changes, and process them when user reloads or online status changes. Handle errors, asking the user when an automatic decision cannot be taken.
`debounceThunk`	Ensure GET requests are not sent too often (e.g. when navigating quickly).
`resolvingSelector`	Declare dependencies of selected data in the stores, and automatically resolve the dependencies upon data selection.
`useDebounceCallback`	Ensure evaluation callbacks are not sent too often (e.g. when editing a form, or when the caching does not fully work).
`validation`	Defines how values are validated.
`useFormValidated`	Publish valid form state to the parent.
`fieldDefinitions`	Powers the `Filter`, `Sort`, `Grid` and `ExcelExport`. Declares the `path` to a specific values, and a `valueDefinition`. The `valueDefinition` provides utils to deal with that value (`compare`, `format`, `render`, `input`).
`parameterDefinitons`	Powers the `Parameters`. Declares grouping, setting keys, default values, input, validation and simulation.
`render`	Defines how values should be rendered.
`Setup`	General setup of the frontend app. Error boundaries which ensure errors do not totally crash the app. Lazy loading per widgets which reduces start-up time. Web worker which eases stratus offline functionality. Authentication and Authorization using `keycloak`. Routing which reads out and modifies the URL. Translations integrating with `phrase`.
`Technical History`	Of impactful entities (those with a version id) changes are stored with corresponding author and timestamp. This helps to retrace who changed what and when.
`Import` / `Export`	Scripts to handle stratus data.
`localdev`	Run stratus locally.

Generic Functionality

Functionality not built for a specific entity; hence easily transferable to other entities.

Functionality	Features
`Remark`	Add a remark to some entity. Includes a short text, the author and the last modify date.
`Images`	Set an image.
`i18nValues`	Set translation of a text value.

Objects

Widgets with objects as the primary resource.

Widget & Purpose	Features
`S7ObjectsOverviewWidget` Compare and find objects.	`DataGrid` and `Filter` with all objects and all properties. An entry can be selected and then appears in the side bar. Actions: `ExcelExport` of the filtered list Sidebar: `S7ObjectDetailsSidebarWidget` shows image and address of a single object and allows to navigate to its master data.
`S7ObjectMasterDataWidget` View & edit properties of object.	Form with all properties grouped into identification, monetary values, description, dimensions and object groups. Actions: Start edit mode. Remove object. Show help. Sidebar: `S7ObjectPictureSidebarWidget` shows image of a single object and allows to replace it.
`S7ObjectPlanningWidget` Estimate future cost based on assessment of single object. Take action by creating a project.	Table with all constructional components (Z/N now, stratus recommended repair year & repair cost). Filter by type of components (shell, services, interior). Show flag of projected components.

	Graph visualizing stratus recommended maintenance and repair cost over the next 20 years. Synchronize highlights with table. Project creation mode: Select unprojected components, then create a project with name and end date. End date must be in the future. Show a warning if the stratus recommended project start is in the past. Highlight selected components in the graph. Actions: Export project evaluation. Export graph image. Show help about stratus recommendations.
`S7ObjectComponentsWidget` Edit components.	If no components configured yet, choose a precomposed set (e.g. school, factory). Add additional components and remove existing ones. Change portion per component. View stratus calculated repair cost and total of portions. View if component is part of a project. Order of the components is defined by the precomposed sets. Actions: Remove all components.
`S7ObjectAssessmentsOverviewWidget` Navigate between assessments and overview.	View and select the latest assessments per component (`S7ObjectLatestAssessmentsWidget`) or an individual assessment (`S7ObjectAssessmentDetailsWidget`). For an individual assessment, the assessment year, date, type (in place, statistical, …) and state (draft, approved, …) is shown. Hidden when other assessment widgets are in edit mode.
`S7ObjectLatestAssessmentsWidget` View latest assessment per component.	See for each component the latest assessment (type, year, Z/N), and its stratus calculated repair cost and repair year. See if component is part of a project. View portion total. Actions: Create new assessment (select components first). Edit components. Download assessment sheet.
`S7ObjectAssessmentDetailsWidget` View and edit a specific assessment.	View all components, either assessed, or not removed (hence removed but assessed components are shown). View components not part of the assessment. View if component is part of a project. For components part of the assessment which are not removed, view stratus calculated repair cost and repair year. View portion total, relative to portion part of the assessment. Edit mode: View stratus calculated Z/N in the assessment year based on the approved assessments. Edit component technical descriptions, and resistance/strain. Add and remove assessed components. Removed components can only be removed from the assessment (in particular, not edited). Actions: Start edit mode. Approve assessment. Edit components. Download assessment sheet. Remove assessment.
`S7ObjectTechnicalDescriptionSidebarWidget` Show technical descriptions of components.	List all components with their technical description. Highlights and scrolls to component if it is selected in `S7ObjectLatestAssessmentsWidget` or `S7ObjectAssessmentDetailsWidget`. Hidden when other assessment widgets are in edit mode.
`S7ObjectHistoryOverviewWidget` View history entries of object.	`DataGrid` with all history entries (project end, title, repair cost, value increasing cost, value neutral cost, maintenance cost). An entry can be selected and then appears in the side bar. Actions: Create history entry. Export filtered list. Sidebar: `S7ObjectHistorySidebarWidget` shows start / end of history entry, the measures (with cost and intervention depth), and a summary of cost per type. Further, it shows the description of the entry.
`S7ObjectHistoryDetailsWidget` Edit history entry of object.	Form with title, project start / end and description. Editable table with entry, intervention depth, repair cost, value increasing cost, value neutral cost, maintenance cost. Entries can be added to the table of type “component” (so title of entry equals constructional component title) or “measure” (so title of entry can be chosen freely). Totals are shown in the table.

Livecycle

Widgets with forecast.

Widget & Purpose	Features
`S7LivecycleCostWidget`	Bar chart visualizing stratus calculated maintenance and repair cost over the next 50 years; with the annuity cost as an average line. Configure start year, duration, inflation and interest rate can be configured. `Filter` over object fields and the constructional component. `DataGrid` with constructional components and their repair year. Sidebar: `S7LivecycleCostSidebarWidget` shows per year either all maintenance cost or all repair cost, depending on which bar is selected in the chart.
`S7LivecycleConditionWidget`	Chart visualizing condition (y-axis) per insurance value (x-axis) in a specific year; with the average condition shown as a line. Color is determined by condition (good, ok, bad …). Configure whether the year shown, and whether the chart considers planned projects. `Filter` over object fields. `DataGrid` with objects, their condition, current value and insurance value. Sidebar: `S7ObjectDetailsSidebarWidget`.

Investment Planner

Plan Projects.

Widget & Purpose	Features
`S7ProjectsOverviewTableWidget`	`DataGrid` with all projects, the buildings they belong to, the project start/end, who created it, and the project cost. The filter allows to filter on these fields. Actions: Export project evaluation; with the same order as currently shown in the UI. Sidebar: `S7ProjectsSidebarWidget` shows the selected project; which building it belongs to, the project end and who created it. A list of the high-level cost follows (cost per projected component and project-level cost position). It allows to enter the detail view of the project.

`S7ProjectsOverviewPlanningWidget`	Table with all basic project info (name, building it belongs to, cost) to the left, sticky. To the right, a grid with years and quarters, and a visualization of when which project phase is active & their cost. The header of the grid shows the cost in that year. `Filter` and `Sort` over shown fields. Actions: Export planning report; with the same order as currently shown in the UI. Sidebar: `S7ProjectsSidebarWidget`.
`S7ProjectDetailsWidget`	Projected constructional component with description, cost (stratus-calculated or user-overridden) and condition at end of project. Further, the calculated condition at the end of the project, as well as when stratus recommends replacement of the component in years. Additional cost positions on the project, and for each component. Each cost position has title, value-increasing and value-neutral costs. For project-level cost positions, additional a description can be set, and the phase(s) the cost belongs to can be chosen: Either phase 1,2, 5.3 or “the rest”. Actions: Export project evaluation, remove project.
`S7ProjectDetailsWidget`	Phases can be activated/deactivated. Then a length and a break after can be specified. Further, the fee percentage can be set. Based on all this information, the timeline and cost per year of the project is calculated. Actions: Reset planning to stratus-proposal.