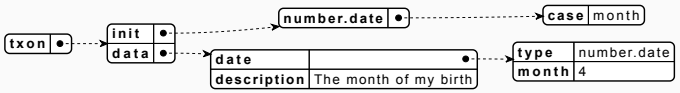
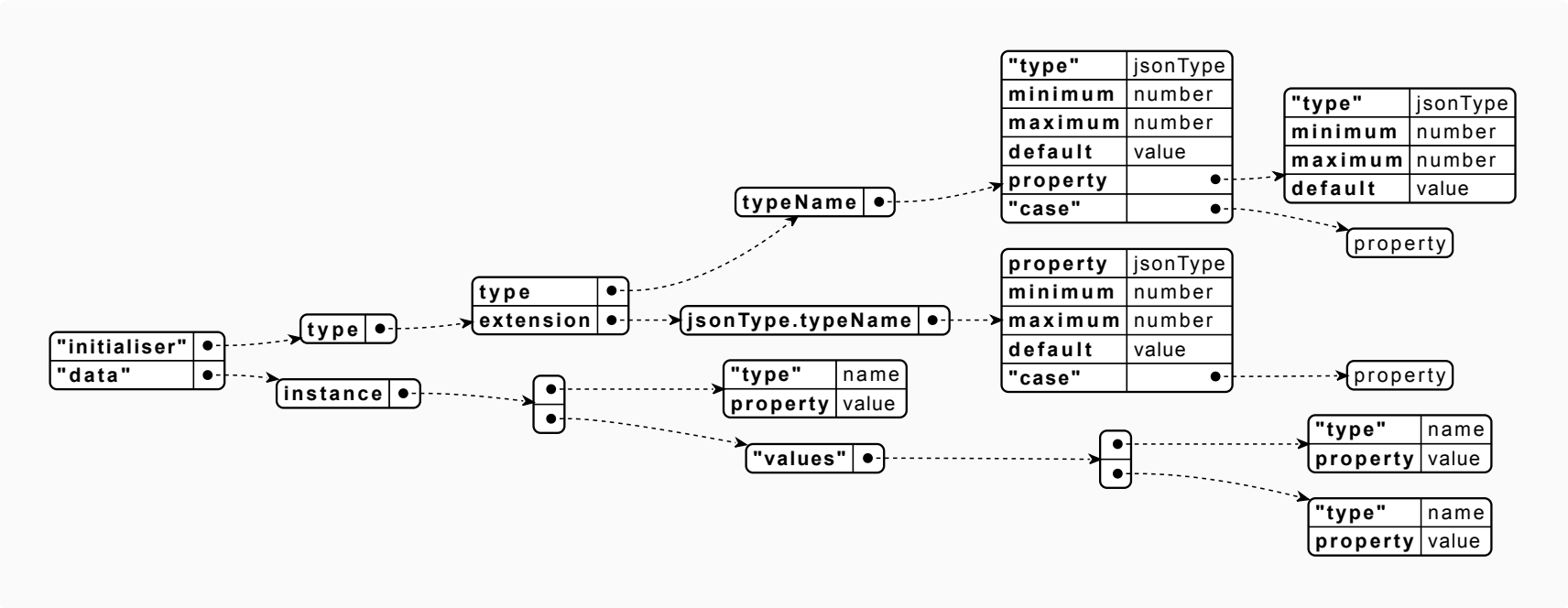
Type-Extensible Object Notation: JSON with Syntax for Types

Grammatical Notation

This project contributes to existing implementations of the JSON specification by proposing a grammar for explicit and extensible typing of values. This proposal is phrased as the `Type-Extensibe Object Notation` (TXON) which is a format that conforms completely to the JSON specification, and as such it maintains full compatibility with existing JSON encoders and decoders. The TXON format is paired with the `TXON.js` library written in JavaScript, to validate the functional implementation and its use in a data structure, by checking conformance to its embedded type system.

A TXON data structure must contain an "init" and "data" property to be validated, but this is not expected to cause issue as JSON structures typically branch from a "data" property at the root node. As such the format is extensibly adding information on types, while maintaining as much of the original structure as possible.

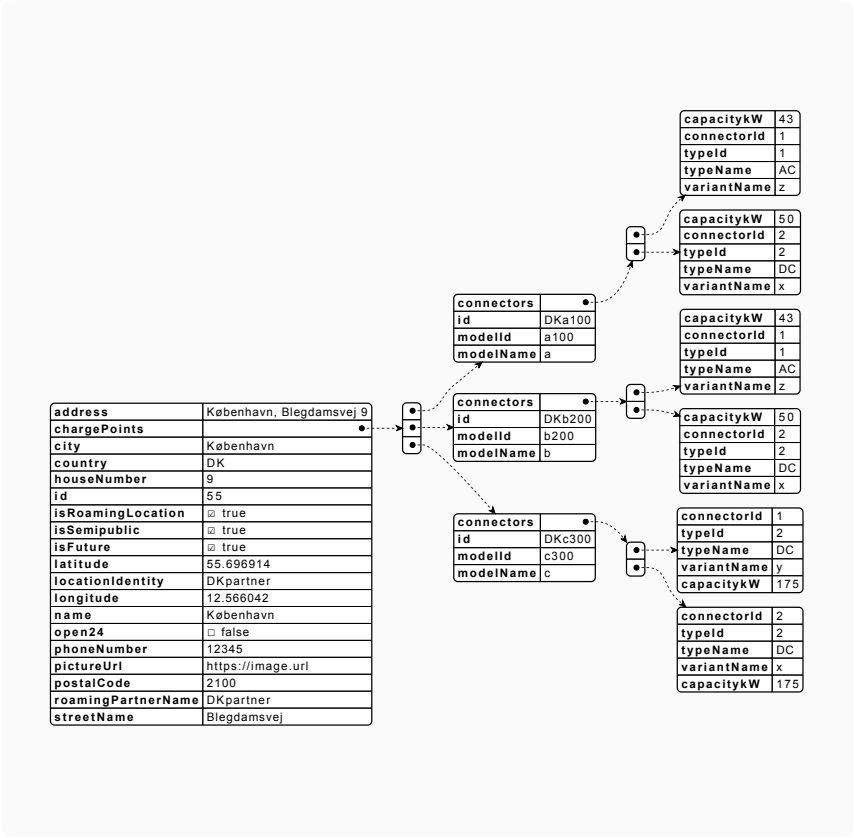



"initialiser"
"data"
type
instance
type
extension
typeName
jsonType.typeName
"type" name
property value
"values"
"type" jsonType
minimum number
maximum number
default value
property
"case"
property
"type" jsonType
minimum number
maximum number
default value
property jsonType
minimum number
maximum number
default value
"case"
property
"type" name
property value
"type" name
property value

Data Structure

Phasellus aliquet convallis arcu in aliquam. Pellentesque lectus orci, vulputate a blandit eget, pretium vel lectus. Aenean ultricies augue at elit rhoncus, quis hendrerit erat egestas. Sed sit amet efficitur sem. Pellentesque euismod, odio eu imperdiet facilisis, mi tortor mattis arcu, a tincidunt sem mauris id est. Maecenas sollicitudin eros ut bibendum sagittis. Suspendisse mattis, risus et luctus lacinia, nisl neque dictum nibh, non hendrerit orci libero non nibh. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia curae; Sed volutpat, ex a sodales semper, turpis mauris accumsan erat, quis malesuada tellus mi non augue.

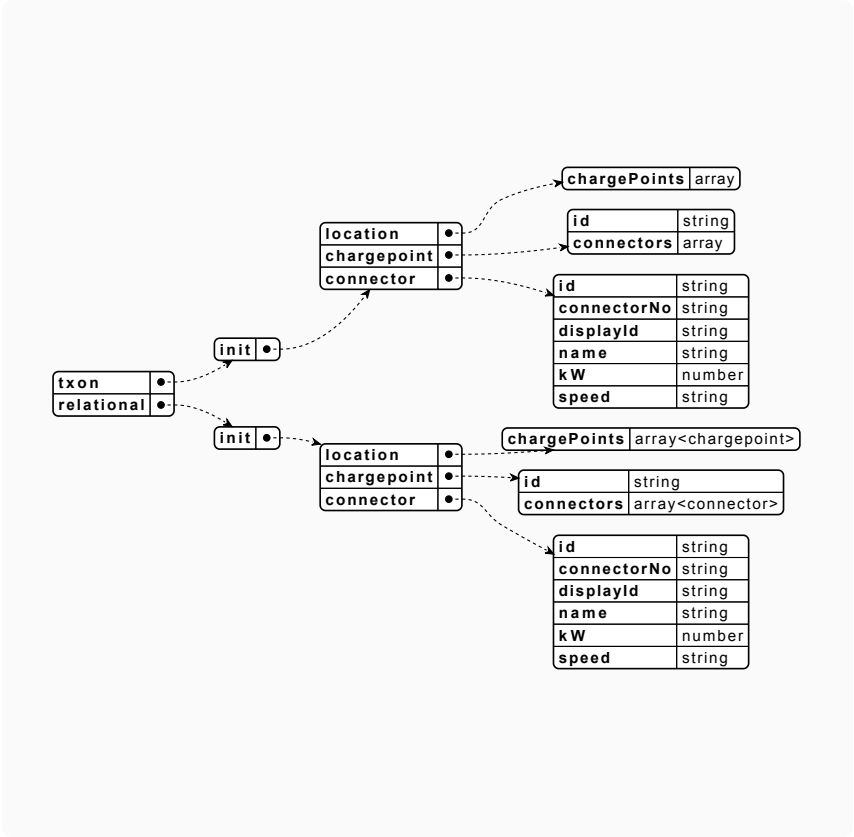
In ultrices dictum dui, quis lacinia mauris hendrerit ac. Donec malesuada viverra elementum. Cras erat turpis, euismod eu tempus vel, cursus non neque. Sed semper tellus bibendum mi consectetur aliquet. Suspendisse interdum auctor turpis sed ultrices. Vestibulum sollicitudin nisi at tellus dictum auctor vel sit amet tellus.

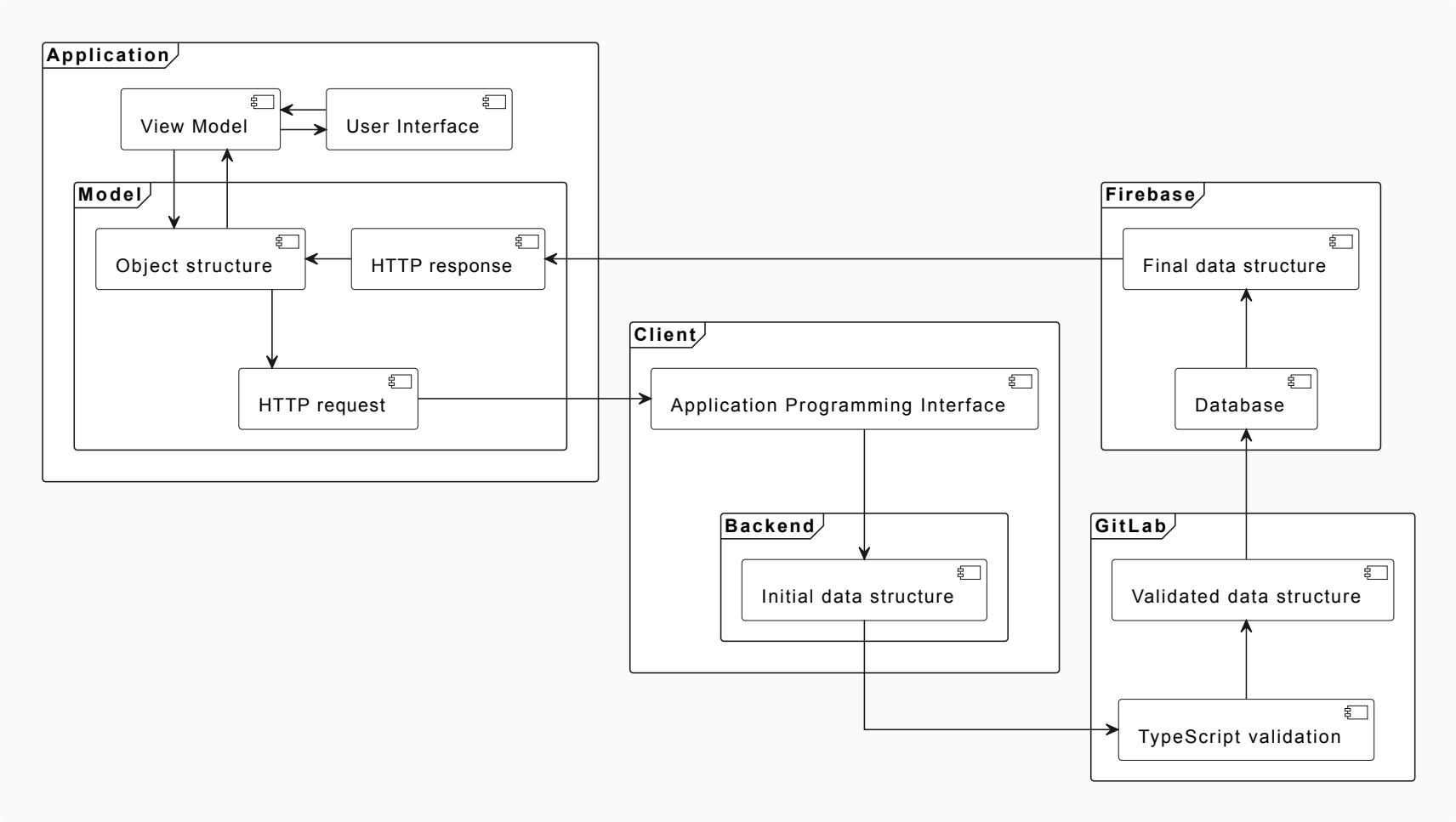


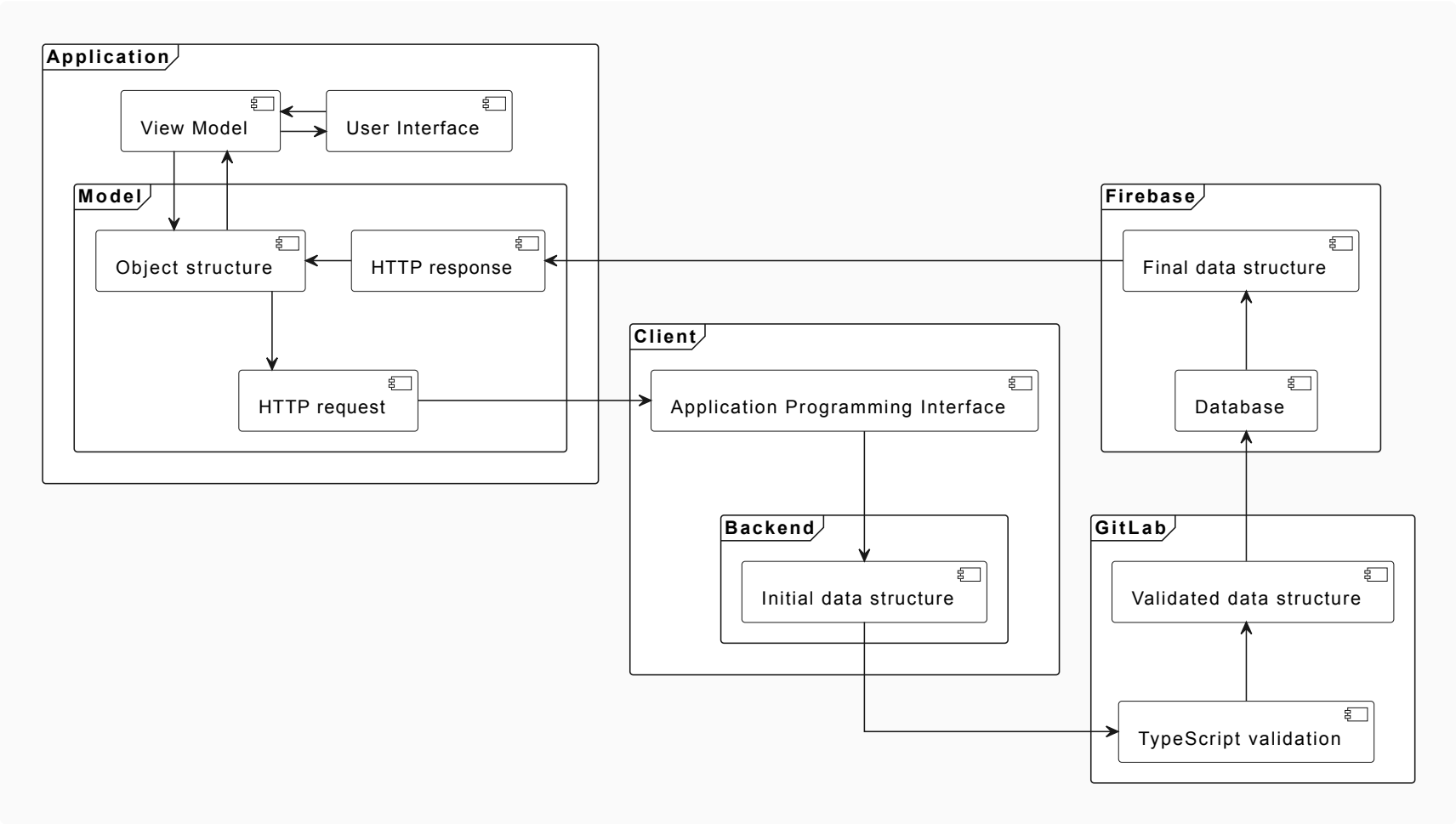
Relational Types

Phasellus aliquet convallis arcu in aliquam. Pellentesque lectus orci, vulputate a blandit eget, pretium vel lectus. Aenean ultricies augue at elit rhoncus, quis hendrerit erat egestas. Sed sit amet efficitur sem. Pellentesque euismod, odio eu imperdiet facilisis, mi tortor mattis arcu, a tincidunt sem mauris id est. Maecenas sollicitudin eros ut bibendum sagittis. Suspendisse mattis, risus et luctus lacinia, nisl neque dictum nibh, non hendrerit orci libero non nibh. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia curae; Sed volutpat, ex a sodales semper, turpis mauris accumsan erat, quis malesuada tellus mi non augue.

In ultrices dictum dui, quis lacinia mauris hendrerit ac. Donec malesuada viverra elementum. Cras erat turpis, euismod eu tempus vel, cursus non neque. Sed semper tellus bibendum mi consectetur aliquet. Suspendisse interdum auctor turpis sed ultrices. Vestibulum sollicitudin nisi at tellus dictum auctor vel sit amet tellus.




Application
View Model
User Interface
Model
Object structure
HTTP response
HTTP request
Firebase
Final data structure
Database
Client
Application Programming Interface
Backend
Initial data structure
GitLab
Validated data structure
TypeScript validation


Application
View Model
User Interface
Model
Object structure
HTTP response
HTTP request
Firebase
Final data structure
Database
Client
Application Programming Interface
Backend
Initial data structure
GitLab
Validated data structure
TypeScript validation