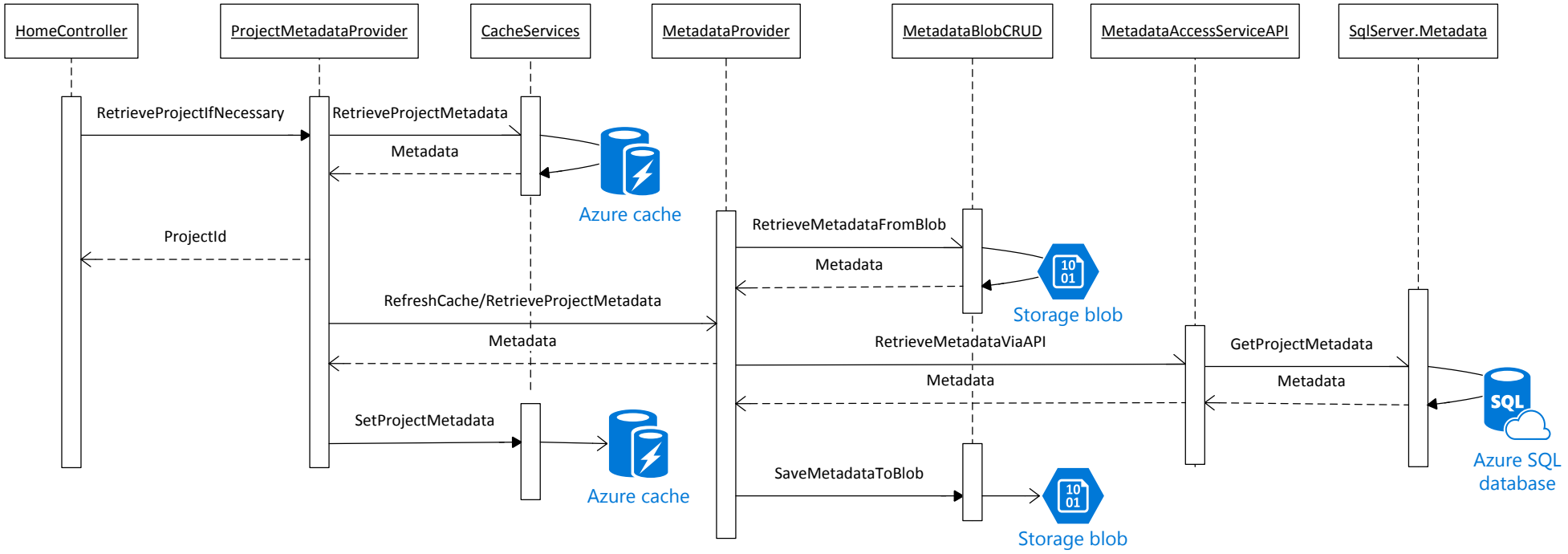
Prime The Metadata Cache



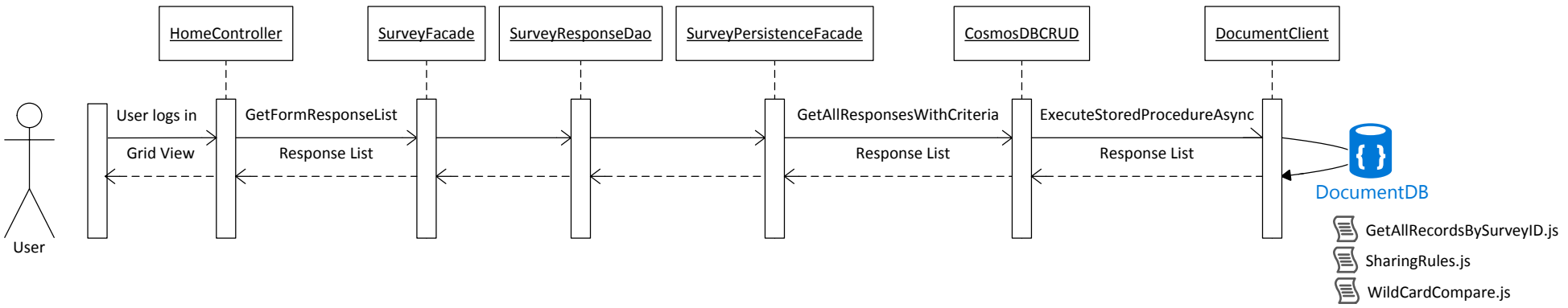
The Home Controller checks if the ProjectId is in Session. If not, the Project Metadata Provider is called at GetProjectId_RetrieveProjectIfNecessary. This call will initiate a loading of cache with project metadata if necessary and return the ProjectId, which is then saved in Session.

The Project Metadata Provider attempts to retrieve the project metadata from Azure Redis Cache. If the metadata is retrieved then the ProjectId is returned to the Home Controller.
If the metadata is not found in cache then the Metadata Provider is called to retrieve the metadata from BLOB storage or Sql Database.
When the project metadata is returned the Cache is refreshed and the ProjectId is returned to the Home Controller.

The Metadata Provider attempts to retrieve the metadata from BLOB Storage. If the metadata is retrieved then the project metadata is returned to the Project Metadata Provider.
If the metadata is not found in BLOB storage then the Metadata Access Service API is called to retrieve the metadata.
When the metadata is returned via the API the metadata is saved to BLOB storage.

The Metadata Access Service API Project Controller calls via the ProjectProxyService to retrieve the project metadata from the Azure SQL database.

Retrieve Responses To Populate Grid



The user logs in and is redirected to the Home Controller. The user is presented with a list of forms that he/she is authorized to interact with.

The user selects a form. If only one form is available then that form will be automatically selected for the user.

The Home Controller uses an instance of the Survey Façade to call GetFormResponseList passing a SurveyAnswerRequest object which defines the criteria to be used to filter and order the list of responses.

The Survey Façade uses an instance of Survey Response DAO to call GetFormResponseByFormId.

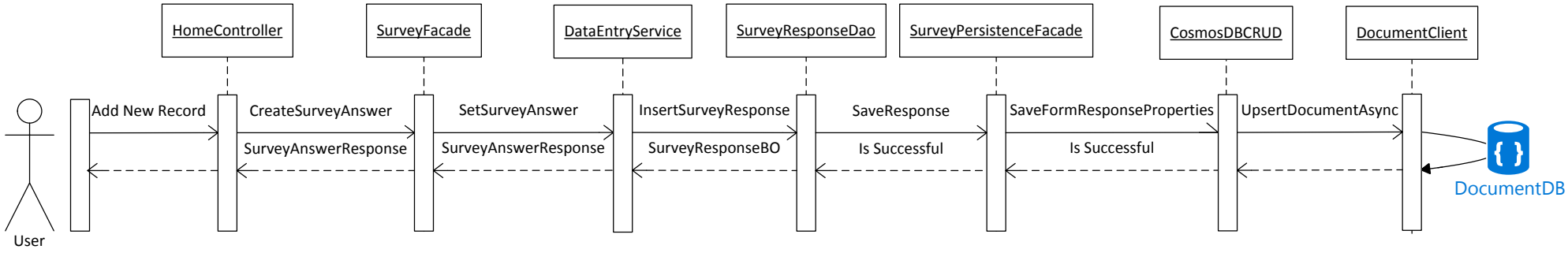
The Survey Response DAO uses an instance of the SurveyPersistenceFacade to call GetAllResponsesWithCriteria.

The Survey Persistence Façade uses an instance of Cosmos DB CRUD to call GetAllResponsesWithCriteria.

A SQL-like query is generated that includes a representation of the query criteria including calls to SharingRules and WildCardCompare JavaScript User Defined Functions to facilitate the more complex filtering requirements.

The `Microsoft Azure Documents Client API ExecuteStoredProcedureAsync` is called to invoke the GetAllRecordsBySurveyID JavaScript stored procedure with the SQL query string passed as a parameter. The resulting list of responses are returned.

Add New Response



The user selects “Add New Record” which invokes the Home Controller which calls the SurveyFacade at CreateSurveyAnswer passing the ResponseContext.

The ResponseContext contains the ResponseId of the new record along with the FormId, UserId, and OrganizationId.

The Survey Façade enlists the SurveyHelper to create the new SurveyAnswerDTO object and FormResponseDetail object from the ResponseContext. A SurveyAnswerRequest is created and the SurveyAnswerDTO object is added to SurveyAnswerRequest.

The DataEntryService is called at SetSurveyAnswer at SetSurveyAnswer passing the SurveyAnswerRequest.

The SurveyAnswerRequest object is transformed into a SurveyResponseBO object and the SurveyResponseProvider is called at InsertSurveyResponse passing the SurveyResponseBO object.

The SurveyResponseProvider passes the call through to the SurveyResponseDao at InsertSurveyResponse.

The SurveyResponseDao passes the SurveyResponseBO to the SurveyPersistenceFacade at SaveResponse which in-turn calls SaveFormResonseProperties which transforms the SurveyResponseBO object into a FormResponseProperties object.

CosmosDBCRUD is called at SaveFormResponseProperties. The `Microsoft Azure Documents Client API UpsertDocumentAsync to persist the record.`