@HBS >An app-assisted approach for the Household Budget Survey

@HBS backend specifications

Version 1

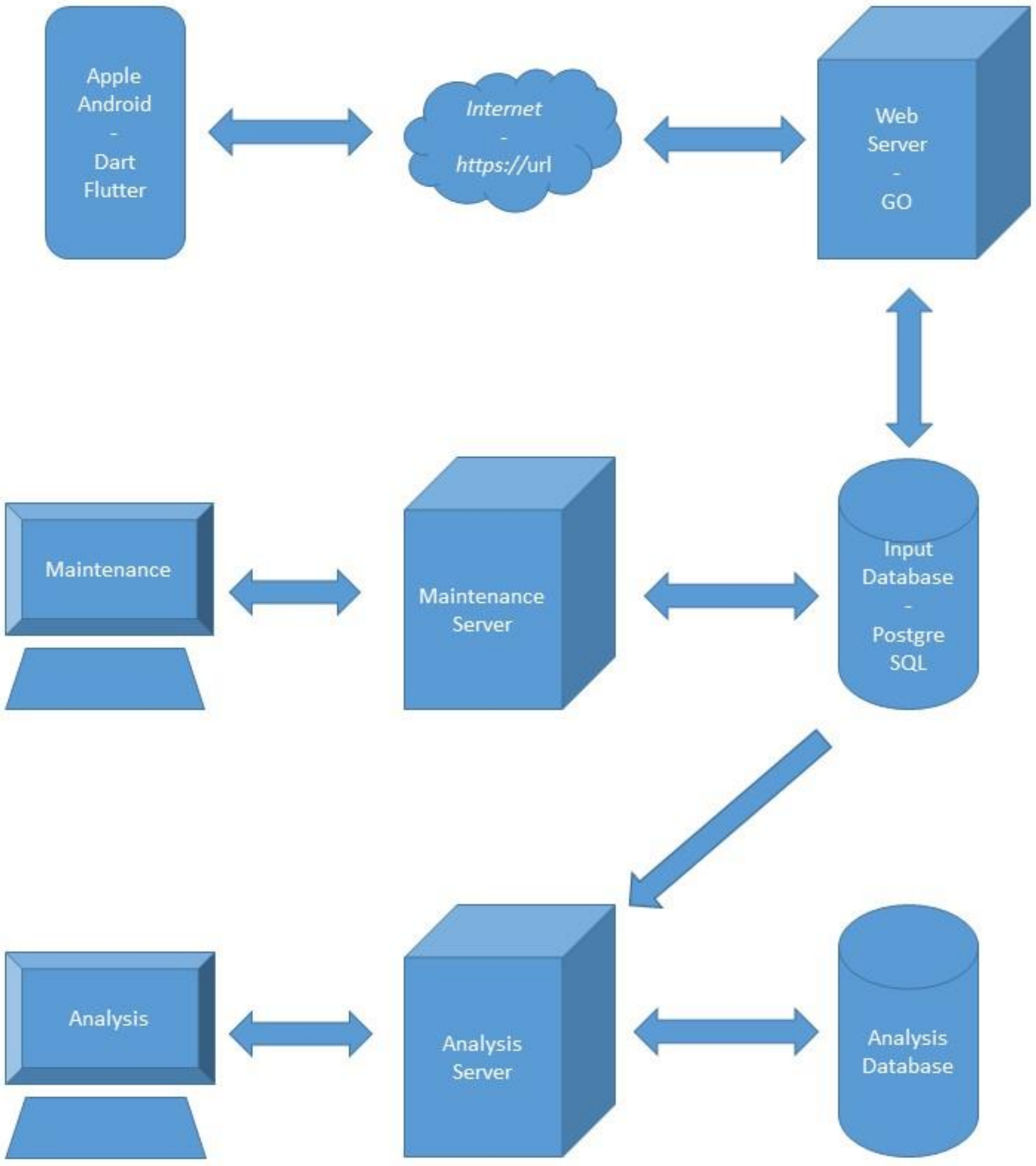
Jack Janssen
Statistics Netherlands
jj.janssen@cbs.nl

project number 847257

February 20, 2020

summary @HBS investigates an app-based approach to the Household Budget Survey. This document describes the backend for the Household Budget Survey app. At the time of writing, the backend was not yet fully operational.

Household Budget Survey - Mobile App - System overview



System parts:

- Users enter/manage data on Apple or Android smartphones with HBS app. HBS app is developed in Dart/Flutter.
- Communication over internet via secure HTTPS URL.
- Web Server monitors HTTPS URL and synchronizes data on smartphones with data in input database. User can manage data on multiple smartphones. Data is timestamped on smartphone, data with latest timestamp is valid. Web server is developed in GO language.
- Input Database stores and synchronizes data from smartphones. Also used for management of usernames and passwords (hashed). Input database is developed in PostgreSQL.

System parts vision (to be implemented in Q2 2020):

- Maintenance and monitoring of user response, management of passwords.
- Maintenance Server for communication between maintenance and input database.
- Analysis of data by statisticians.
- Analysis Server for analysis of data in analysis database, and data import from input database to analysis database.
- Analysis Database for final data storage and data analysis.

Functionality

The main functionality of the HBS backend is:

- User authentication: check whether the user password matches the (hashed) password in the database
- Store receipt data: the data gathered on the smartphone is also stored in the database.
- Synchronize receipt data: keep the data synchronized between smartphones if the user uses multiple smartphones.
- Monitoring: respondent activity and app usage is monitored and may be input to follow-up actions

The HBS-App maintains the receipt data locally on a phone. When the HBS-App is started, or when a receipt is entered or changed the app tries to connect to the GO server to synchronize the locally stored data with the data in the database. The HBS-App does not need a continuous connection with the GO server, when there is a connection all data will be synchronized.

Status

The Dutch Statistics office is in a transition going towards an infrastructure that is delivered through Pivotal Cloud Foundry. We aimed at releasing the HBS backend with this new infrastructure. Unfortunately this infrastructure is not yet fully operational therefore we could not test a full working backend.
We did manage to test the backend on a Pivotal Cloud Foundry playground outside of the Dutch Statistics office.

- the database size on this playground was only 20 MB, which is just enough for uploading 4 photos of receipts.
- the URL we used for this test was not secure (HTTP instead of HTTPS).

The backend was tested on a small scale (one user with two smartphones, max. 4 photos). On this small scale (using a HTTP URL) everything worked fine.

PostgreSQL Input Database

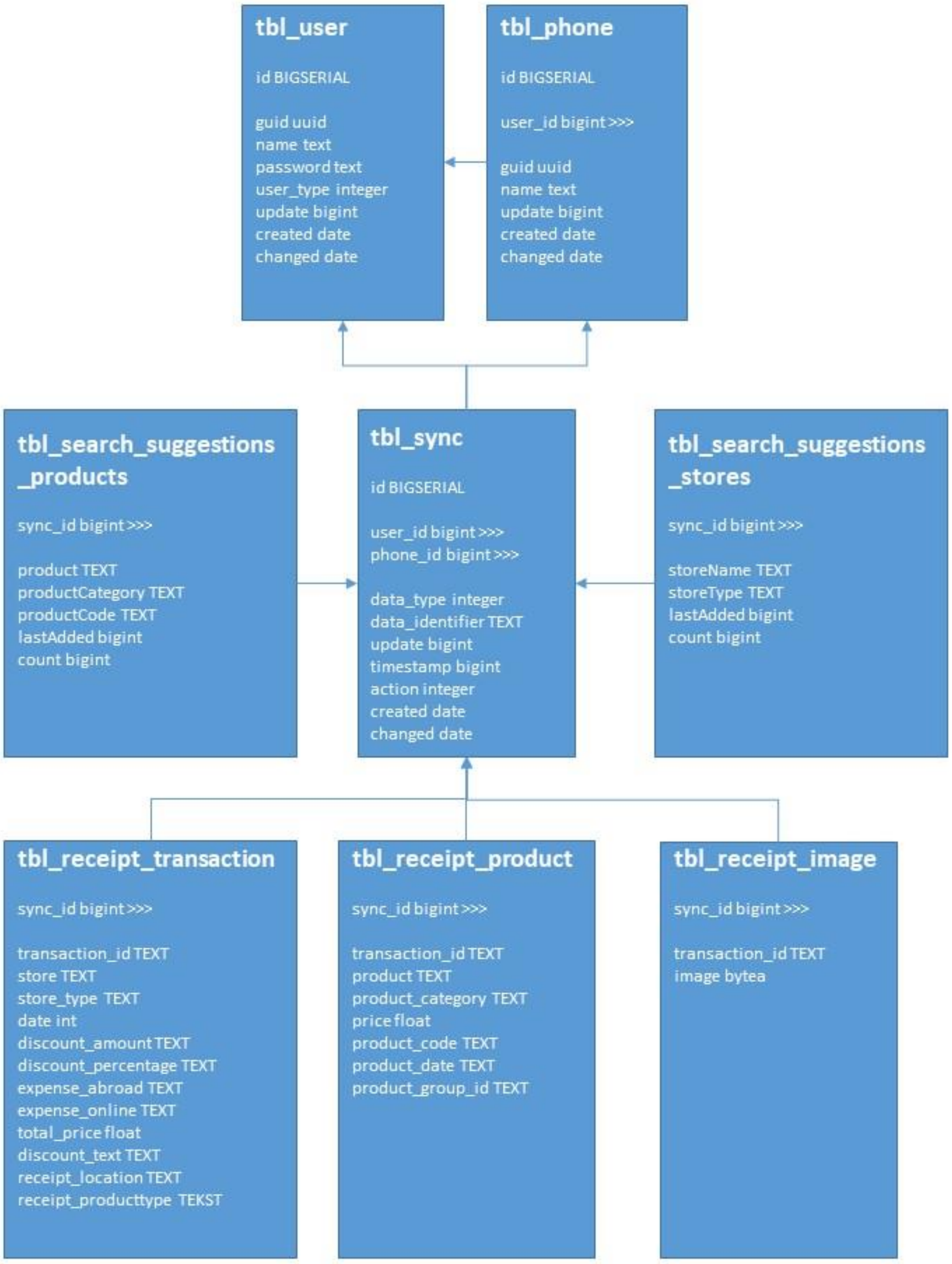
The PostgreSQL Input database contains 8 tables (see image data model below):

• tbl_user	: username + hashed password
• tbl_phone	: multiple phones per users
• tbl_sync	: data synchronization mechanism
• tbl_receipt_transaction	: receipt information
• tbl_receipt_product	: receipt products
• tbl_receipt_image	: receipt photos
• tbl_search_suggestions_products	: frequently used products
• tbl_search_suggestions_stores	: frequently used stores

There is no further functionality in the database. The contents of these tables is managed by the GO server. The creation script for these tables is located in the file structure of the GO-language server. It is possible to initialize the database (add users + hashed passwords) with the GO-language server.

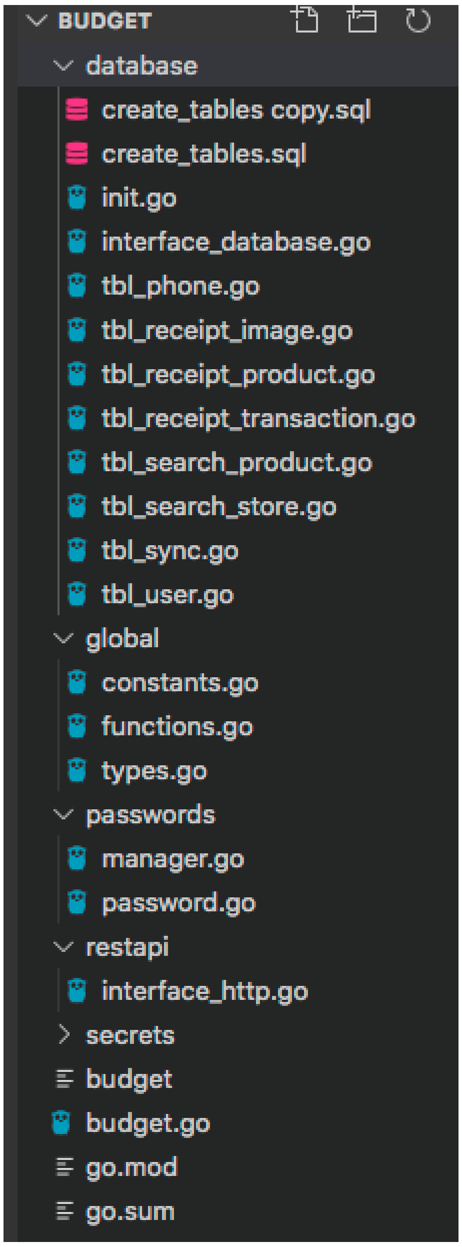
The required size of the database depends on the number of users, the length of the survey period and the number of receipts a user enters per day. Per receipt the required database size is: +/- 5 MB, this is roughly the size of a photo + 1KB of textual data. Example: If 100 users take 2 photos per day for a period of one month, the required database size is: 100 * 2 * 5 MB * 30 = 30.000 MB = 30 GB.

Household Budget Survey Input data Model.



GO-language server

File structure of GO server for the HBS-App:



```
BUDGET
  database
    create_tables copy.sql
    create_tables.sql
    init.go
    interface_database.go
    tbl_phone.go
    tbl_receipt_image.go
    tbl_receipt_product.go
    tbl_receipt_transaction.go
    tbl_search_product.go
    tbl_search_store.go
    tbl_sync.go
    tbl_user.go
  global
    constants.go
    functions.go
    types.go
  passwords
    manager.go
    password.go
  restapi
    interface_http.go
  secrets
  budget
  budget.go
  go.mod
  go.sum
```

- PostgreSQL script for creating tables
- Initial connection with database
- Main interface with database
•
- Standard functions per table (insert, update, delete, select etc.)

- General constants
- General functions
- General Types

- Create initial users + hashed passwords
- Password generation
•
- Rest-api URL Interface
•
- Secrets folder with users + passwords
- Compiled GO server executable
- Main entry of GO server
- Used modules
- Module summary

The main entry point for the GO server budget.go is initialized via environment settings:

• PORT	: http port
• BUDGET_HOST	: database host
• BUDGET_DBNAME	: database name
• BUDGET_PORT:	: database port
• BUDGET_USER	: database user
• BUDGET_PASSWORD	: database password

These environment settings are required and have to be set for your environment.

The GO server can be used to generate a set of standard usernames and passwords. See the readme section in passwords/manager.go. The generated usernames and hashed password are used to initialize the database. Also, in the folder secrets, files will be generated with usernames and passwords that can be communicated with users.

Future steps (Q2 and Q3 2020)

The HBS backend as it is, is only equipped to authenticate users and store receipt data in an input database. Additional functionality is not yet available.

The not yet implemented part of the system overview is:

- Maintenance and monitoring of user response, management of passwords.
- Maintenance server for communication between maintenance and input database.
- Data analysis by substantive researchers and methodology
- Analysis server for analysis of data in analysis database, and data import from input database in analysis database.
- Analysis database for final data storage and data analysis.

While this functionality is not available the input database can be managed/queried directly with the pgAdmin_4 tool for PostgreSQL databases.

Additional functionality:

- Analyze photos: create fully classified receipts from photos of receipts. This functionality can be added to the web server or (if required) another server with access to the input database.
- App usage data: at this moment we only gather data of receipts. We are interested in:
    - Type of phone used
    - Dwell time on app pages
    - Technical problems/crashes
    - Use of certain help options
    - Consultation of personal statistics in the app