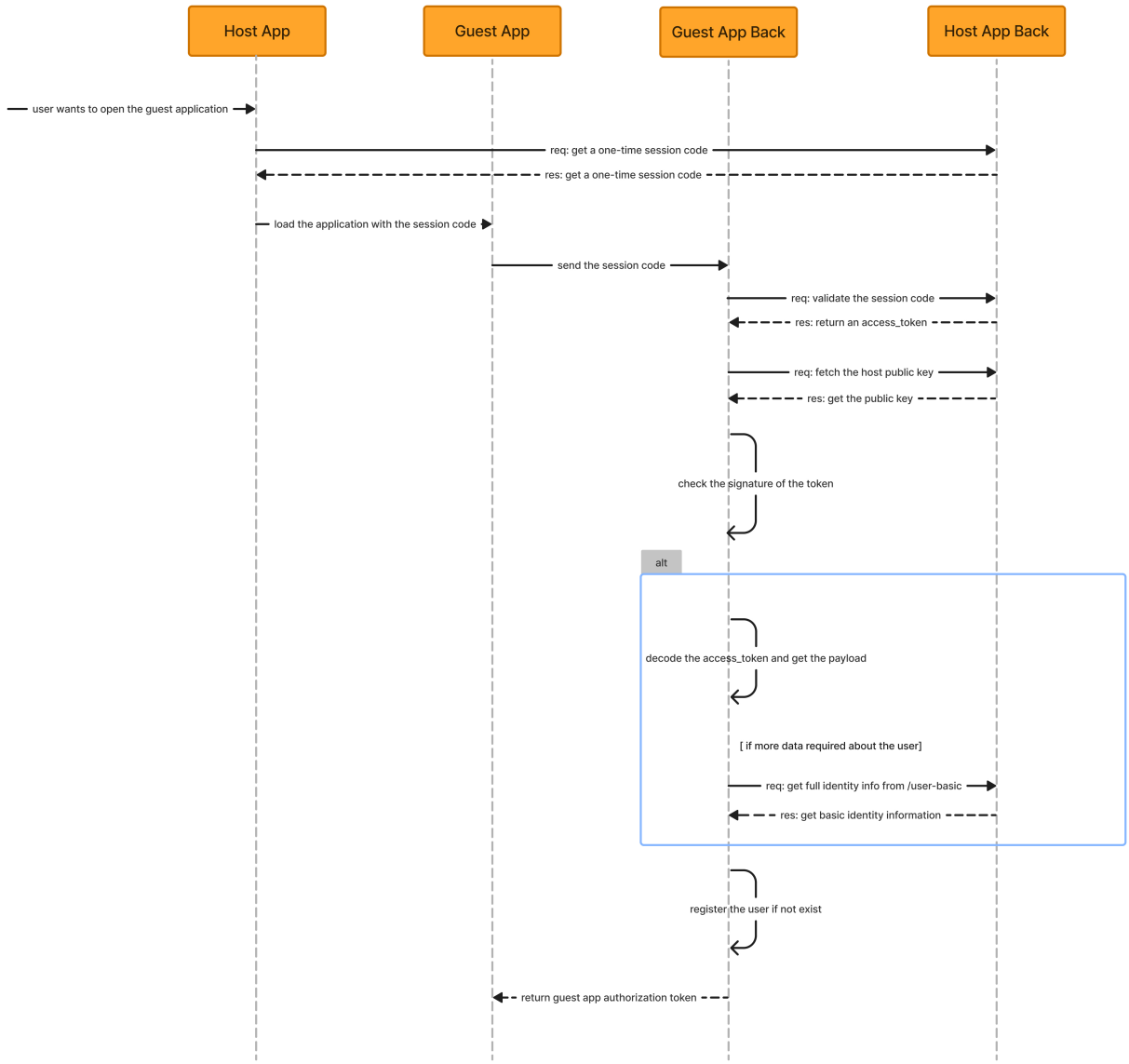
SSO - Native Hosted

The current document present the architecture on how to host different applications in a hosted application which in this document we denote as the **Injast SuperApp**.

Flow

The general flow is depicted in the following diagram.



Detailed Steps

- In the host application, user decides to open the guest application.
- Prior opening the guest application, the host application calls the host backend to retrieve a one-time session code.

This session code is unique to the client, can be used only one-time and has a short-lived expiration window (approximately 15m).

- Host application opens the guest application using the following schema.

```
https://{guest_application}?session_code={session_code}
```

- The guest application sends the session code to its backend server.
- The guest backend sends the session code to the host backend to check whether the session code is valid or not.

The access to the host back is also restricted and whitelisted using the IP address.

- If the session code is valid the host backend returns the following response.

```
{
    "meta": {
        "success": true
    },
    "data": {
        "access_token": "base64 encoded access_token with header, payload and signature"
    }
}
```

This tokens have the following structure:

{header}.{payload}.{signature}

The structure of the payload is as follows:

```
{
        "nid": "0012345678",
    "mbc": "98",
    "mbn": "9121234567",
    "sub": "as6hjfk",
          "iss": "https://{backend_base_url}/",
          "aud": "sso",
          "exp": 1700003600,
          "iat": 1700000000,
          "scp": "update:user-banking read:user-banking read:user-basic",
          "rle": ["guest-app"]
}
```

- Guest backend checks the authenticity of the token's signature.

The guest backend doesn't store the public key manually. Instead, it would be fetched dynamically from the host backend's JWKS (JSON Web Key Set) endpoint.

https://{backend_base_url}/.well-known/jwks.json

- The guest backend stores the access token and bind it to the user based on the national_id field (nid) in the payload.
- As long as the access token is valid, and the guest has sufficient permissions, the guest backend can inquiry various information about the user.
- If the guest backend receives the required information, it will generate an authorization token (designed by the guest application), and return it to the guest application to authenticate the user.

Service Specs

/sso/exchange-session-code

Method: POST

URL: https://{backend_base_url}/service/user/sso/exchange-session-code

Request Body:

key	description	example
session_code	the session code is obtained as a query param in the guest application	2fcc8778-222d-4618-ad30-75a9b0d6fb88

Response Schema:

```
{
    "meta": {
        "success": true
    },
    "data": {
        "access_token": "base64 encoded access_token with header, payload and signature"
    }
}
```

```
{
   "meta": {
       "success": false,
       "error_code": “InternalError”,
       "error_help": ”https://injast.life/help/errors/InternalError”
   },
   "message": “something went wrong!”
}
```

Response Body:

key	description	example
meta > success	the status of the request, if its false the request is failed and the error code returns code in the `meta > error_code` and the message in `message`	true
meta > error_code	if the request fails, the error code represents the code of the failure	NotAuthenticated
message	if the request fails, the message represents a description about the error	the given credential is incorrect!
data > access_token	if the request is successful, the access token returns, which must be stored and used in the next request to fetch data. this token is specific for the user and must be maintained and binded to the user until is valid.	

/sso/user-basic

Method: GET

URL: https://{backend_base_url}/service/user/sso/user-basic

Request Header:

key	description	example
Authorization	the access token which is gained from the exchange session code step	

Response Schema:

```
{
    "meta": {
        "success": true
    },
    "data": {
        "national_id" : "",
        "first_name": "",
        "last_name": " ",
        "birthdate": "1991-04-12",
        "postal_code":"1234567890",
        "email": "username@injast.life",
        "mobile_number" : "9123332222",
        "mobile_country_code" : "98"
    }
}
```

```
{
   "meta": {
       "success": false,
       "error_code": “InternalError”,
       "error_help": ”https://injast.life/help/errors/InternalError”
   },
   "message": “something went wrong!”
}
```

Response Body:

key	description	example
meta > success	the status of the request, if its false the request is failed and the error code returns code in the `meta > error_code` and the message in `message`	true
meta > error_code	if the request fails, the error code represents the code of the failure	NotAuthenticated
message	if the request fails, the message represents a description about the error	the given credential is incorrect!

/sso/user-banking

Method: GET

URL: https://{backend_base_url}/service/user/sso/user-banking

Request Header:

key	description	example
Authorization	the access token which is gained from the exchange session code step	

Response Schema:

```
{
    "meta": {
        "success": true
    },
    "data": {
        "accounts": [
                        {
                        "pan": "6362147010005732",
                                "iban": "IR330620000000202901868005",
                                "account_number": "0202901868005",
                                "bank": "pasargad"
                        }
                ]
    }
}
```

```
{
   "meta": {
       "success": false,
       "error_code": “InternalError”,
       "error_help": ”https://injast.life/help/errors/InternalError”
   },
   "message": “something went wrong!”
}
```

Response Body:

key	description	example
meta > success	the status of the request, if its false the request is failed and the error code returns code in the `meta > error_code` and the message in `message`	true
meta > error_code	if the request fails, the error code represents the code of the failure	NotAuthenticated
message	if the request fails, the message represents a description about the error	the given credential is incorrect!
data > accounts	list of all verified bank accounts that belongs to the user	
data > accounts > pan	the card number of the account	
data > accounts > iban	represents the sheba of the account	
data > accounts > bank	a unique code representing the bank that issued the account	