**BabyCrab**

Table of Contents

[I. Project structure 1](#_Toc89074726)

[II. Workflow 3](#_Toc89074727)

[III. Technologies 3](#_Toc89074728)

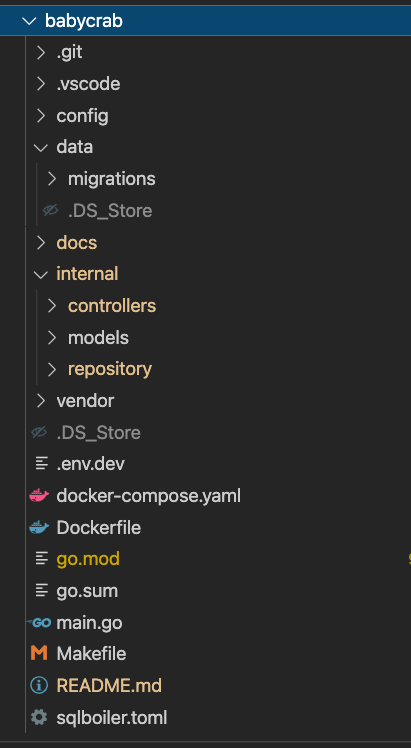
[IV. Setup application 3](#_Toc89074729)

[V. Running 3](#_Toc89074730)

[VI. Teardown application 4](#_Toc89074731)

[VII. Api 5](#_Toc89074732)

1. **Project structure**



1. **Workflow**

Request => Controller => Repository => model => Database

* Controller: Get request from http request, decode, validate, call controller, write http response.
* Controller: Handler business logic, call repositories.
* Model: Data access layer.

1. **Technologies**

* Golang: go module, go-chi, sqlboiler, db-migration, mockery
* Postgres
* Docker
* Makefile

1. **Setup application**

* Make sure that your Docker is up and running
* From folder `babycrab/`, run command `make setup`

1. **Running**

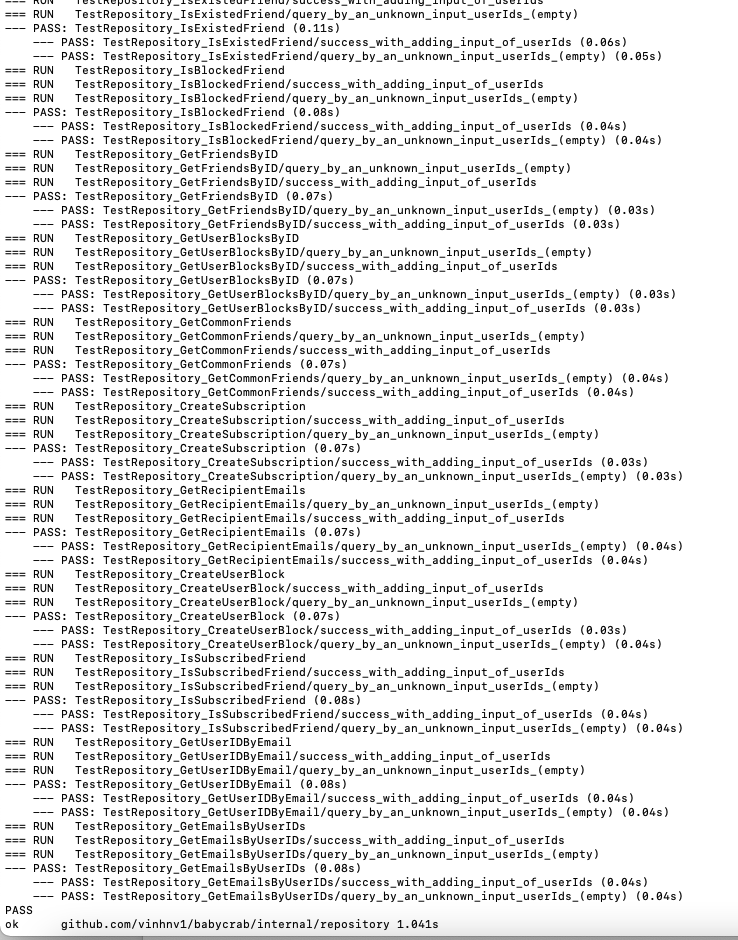
* Run app:

Start the server with command `make run`

Server running on: `https://localhost:8080`

* Run test:

Run unit test with commnd `make test`



1. **Teardown application**

* From folder `babycrab/`, run command `make teardown`

1. **List APIs**
2. **Create friend**

POST: http://localhost:8080/v1/friends

Parameter request:

{

"friends": [

"andy@example.com",

"john@example.com"

]

}

1.1 Success:

Status code: 200 OK

{

"success": true

}

1.2 Error: friendship is existing

Status code: 500 Internal Server

{

"message": "The friend relationship has been existed",

"success": false

}

1.3 Error: user is not existing

Status code: 500 Internal Server

{

"message": "test@example.com is not exists",

"success": false

}

1.4 Error: users have blocked by each other

Status code: 500 Internal Server

{

"message": "The users have blocked each other",

"success": false

}

1. **List Friends**

GET-http://localhost:8080/v1/friends

Parameter request:

{

"Email":"andy@example.com"

}

2.1 Success

Status code: 200 OK

{

"count": 1,

"friends": [

"lisa@example.com"

],

"success": true

}

2.2 Error: user is not existing

Status code: 500 Internal Server

{

"message": "test@example.com is not exists",

"success": false

}

1. **Get common friends**

GET - http://localhost:8080/v1/commonFriends

Parameter request:

{

"friends": [

"andy@example.com",

"john@example.com"

]

}

3.1 Success

Status code: 200 OK

{

"count": 1,

"friends": [

"common@example.com"

],

"success": true

}

3.1 Error: user is not existing

Status code: 500 Internal Server

{

"message": "test@example.com is not exists",

"success": false

}

1. **Create subscription**

POST - http://localhost:8080/v1/subscription

Parameter request:

{

"requestor": "andy@example.com",

"target": "lisa@example.com"

}

4.1 Success

Status code: 200 OK

{

"success": true

}

4.2 users have blocked by each other

Status code: 500 Internal Server

{

"message": "The users have blocked each other",

"success": false

}

4.3 users have subscribed by each other

Status code: 500 Internal Server

{

"message": "The users have subscribed each other",

"success": false

}

1. **Create user block**

POST - http://localhost:8080/v1/blocking

Parameter request:

{

"requestor": "common@example.com",

"target": "kate@example.com"

}

5.1 Success

Status code: 200 OK

{

"success": true

}

5.2 users have blocked by each other

Status code: 500 Internal Server

{

"message": "The users have blocked each other",

"success": false

}

1. **Get Recipients**

GET - http://localhost:8080 /v1/recipients

Parameter request:

{

"sender": "lisa@example.com",

"text": "Hello World! kate@example.com"

}

6.1 Success

Status code: 200 OK

{

"recipients": [

"common@example.com",

"kate@example.com"

],

"success": true

}

6.2 sender is not existing

Status code: 500 Internal Server

{

"message": "test@example.com is not exists",

"success": false

}