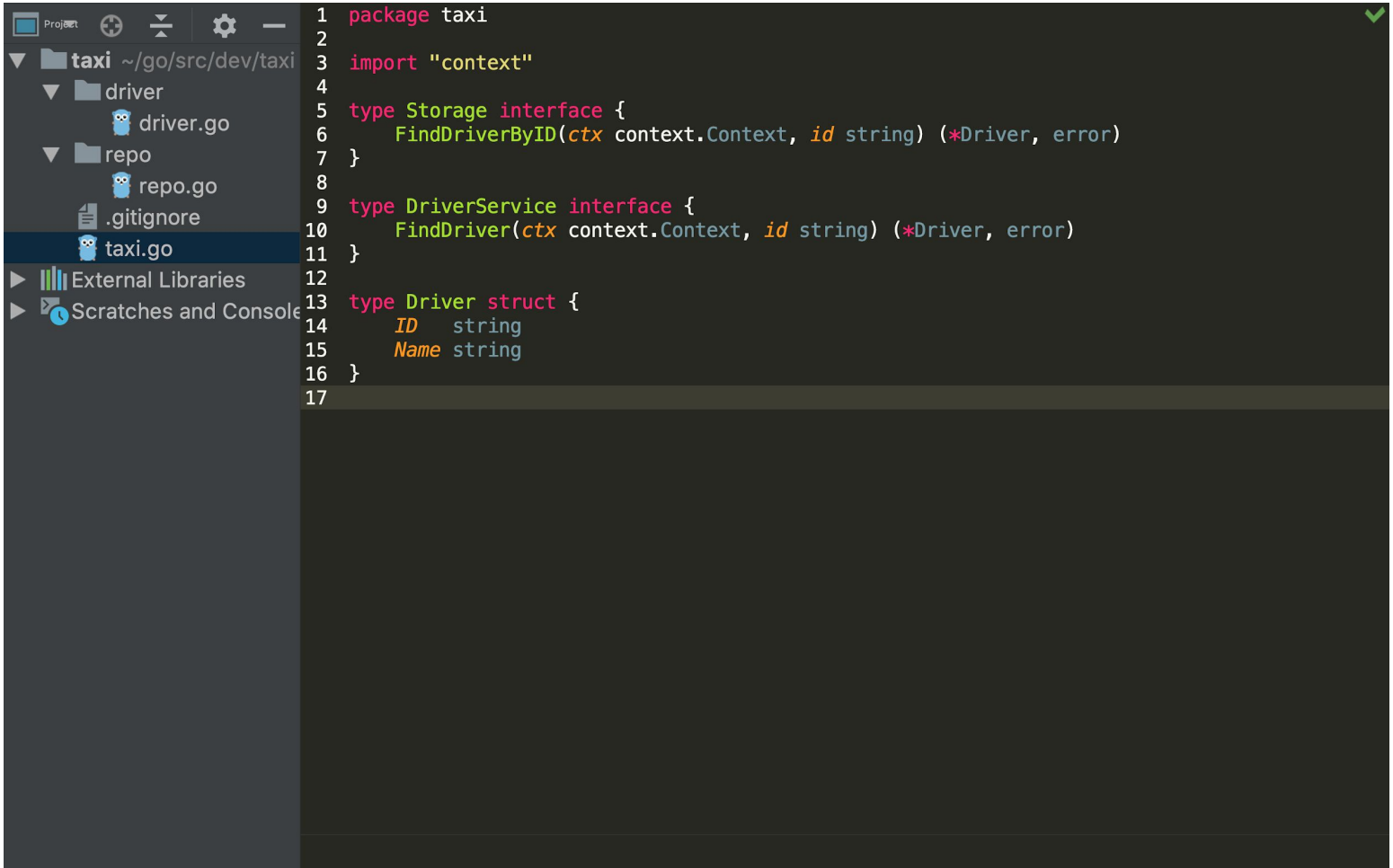
Golang generator of client to sql database based on interface

Zaur Abasmirzoev | 27.06.2019

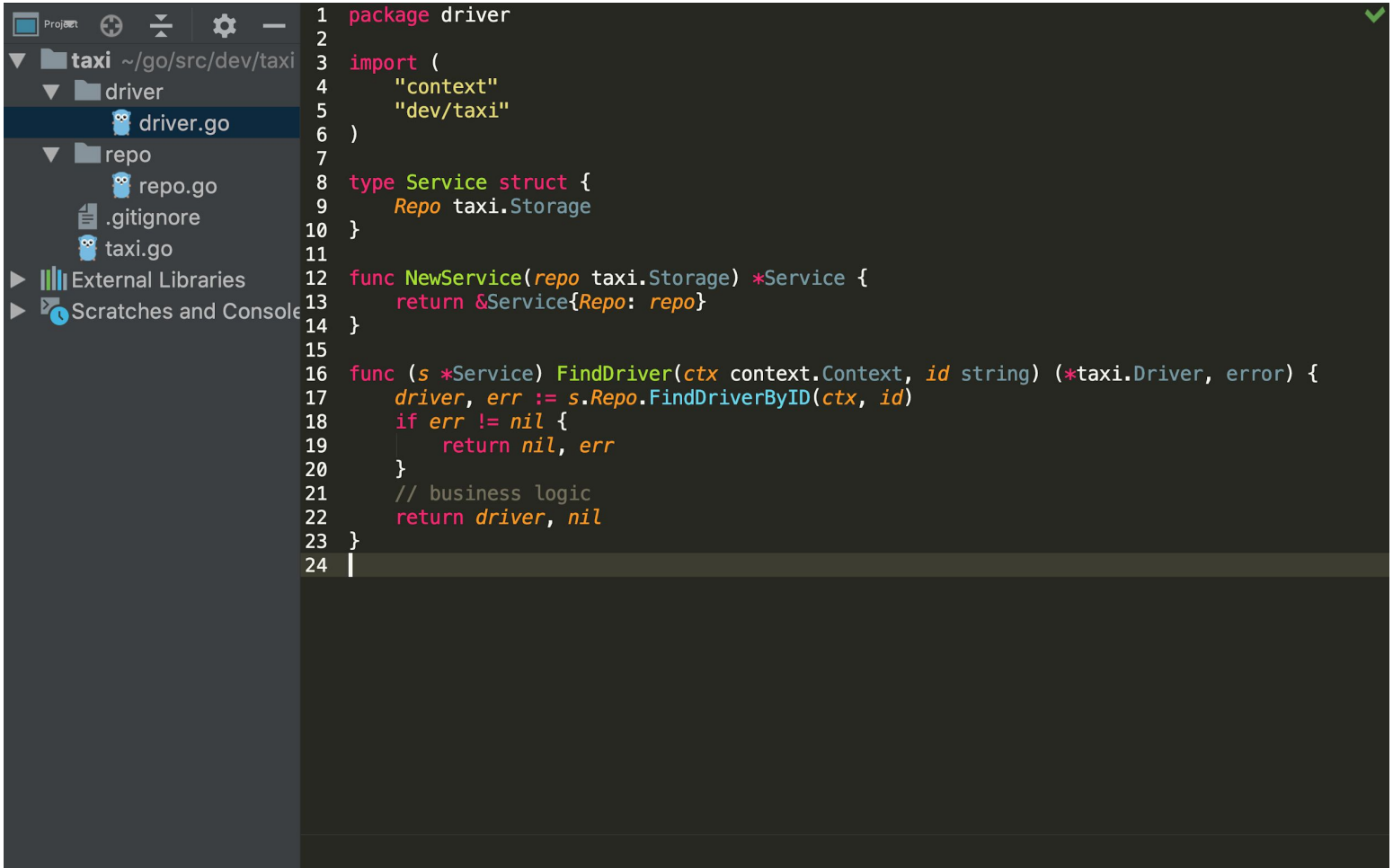
Gett

Repository pattern



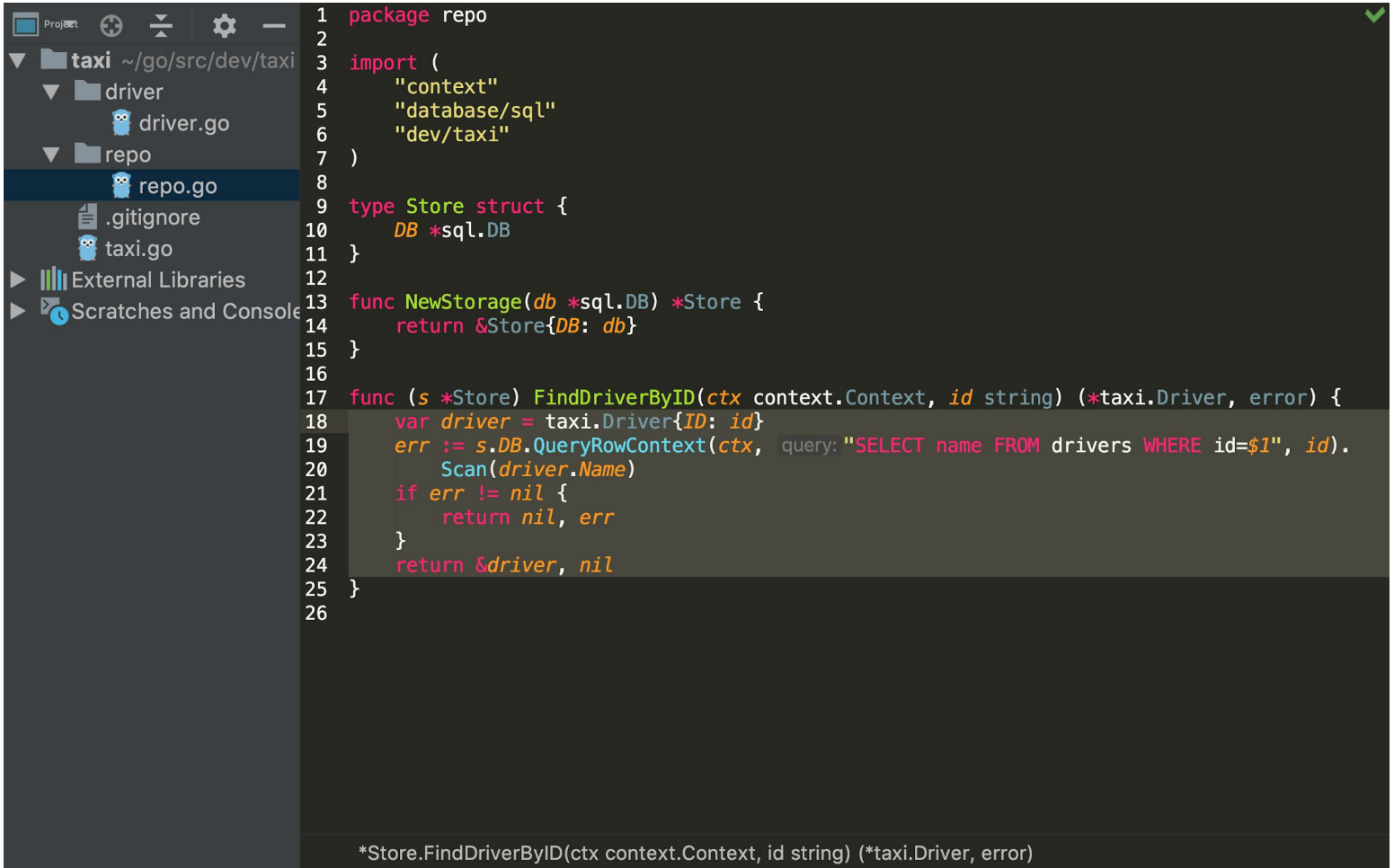

```go
package taxi

import "context"

type Storage interface {
    FindDriverByID(ctx context.Context, id string) (*Driver, error)
}

type DriverService interface {
    FindDriver(ctx context.Context, id string) (*Driver, error)
}

type Driver struct {
    ID   string
    Name string
}
```

Repository pattern




```go
package driver

import (
    "context"
    "dev/taxi"
)

type Service struct {
    Repo taxi.Storage
}

func NewService(repo taxi.Storage) *Service {
    return &Service{Repo: repo}
}

func (s *Service) FindDriver(ctx context.Context, id string) (*taxi.Driver, error) {
    driver, err := s.Repo.FindDriverByID(ctx, id)
    if err != nil {
        return nil, err
    }
    // business logic
    return driver, nil
}
```

Repository pattern



Requirements

1. **Описание взаимодействия в виде интерфейса.**
2. **Интерфейс описывается методами и сообщениями запросов и ответов.**

Запрос

- SQL строка "SELECT name FROM drivers WHERE id=$1"
- Аргументы запроса

```
taxi.FindReq{
        ID: "d243b7e2-8833-4475-b6f7-70998918203b"
}
```

Ответ

```
taxi.Driver{
        ID:   "d243b7e2-8833-4475-b6f7-70998918203b",
        Name: "Ivan Sokol",
}
```

Requirements

3. **Поддержка связывания переменных и подготовленных выражений (prepared statements).**

```
stmt, err := db.PrepareContext(ctx, "SELECT name FROM drivers WHERE id=$1")
row = stmt.QueryRowContext(ctx, id)
```

Requirements

4. **Поддержка именованных аргументов.**

```
db.ExecContext(ctx, `DELETE FROM orders WHERE created_at < @end`,
sql.Named("end", endTime))
```

```
db.ExecContext(ctx, `DELETE FROM orders WHERE created_at < $1`, endTime)
```

Requirements

5. Связывание ответа базы данных с полями структуры данных сообщения.

```
zaur@taxi > SELECT * FROM drivers;
                id                                    |    name
--------------------------------------------------------+-----------
 d243b7e2-8833-4475-b6f7-70998918203b | Ivan Sokol
(1 row)


type Driver struct {
        ID   string
        Name string
}


taxi.Driver{
        ID:  "d243b7e2-8833-4475-b6f7-70998918203b",
        Name: "Ivan Sokol",
}
```

Requirements

6. **Поддержка нетипичных структур данных (массив, json).**

```
db.Query(`SELECT * FROM t WHERE id = ANY($1)`, pq.Array([]int{235, 401}))

var x []sql.NullInt64
db.QueryRow('SELECT ARRAY[235, 401]').Scan(pq.Array(&x))
```

Requirements

6. **Прозрачная работа с транзакциями.**

```
store := NewStorage(db)

tx := store.Begin()
tx.CreateDriver(...)
tx.UpdateOrder(...)
tx.Commit()
```

Requirements

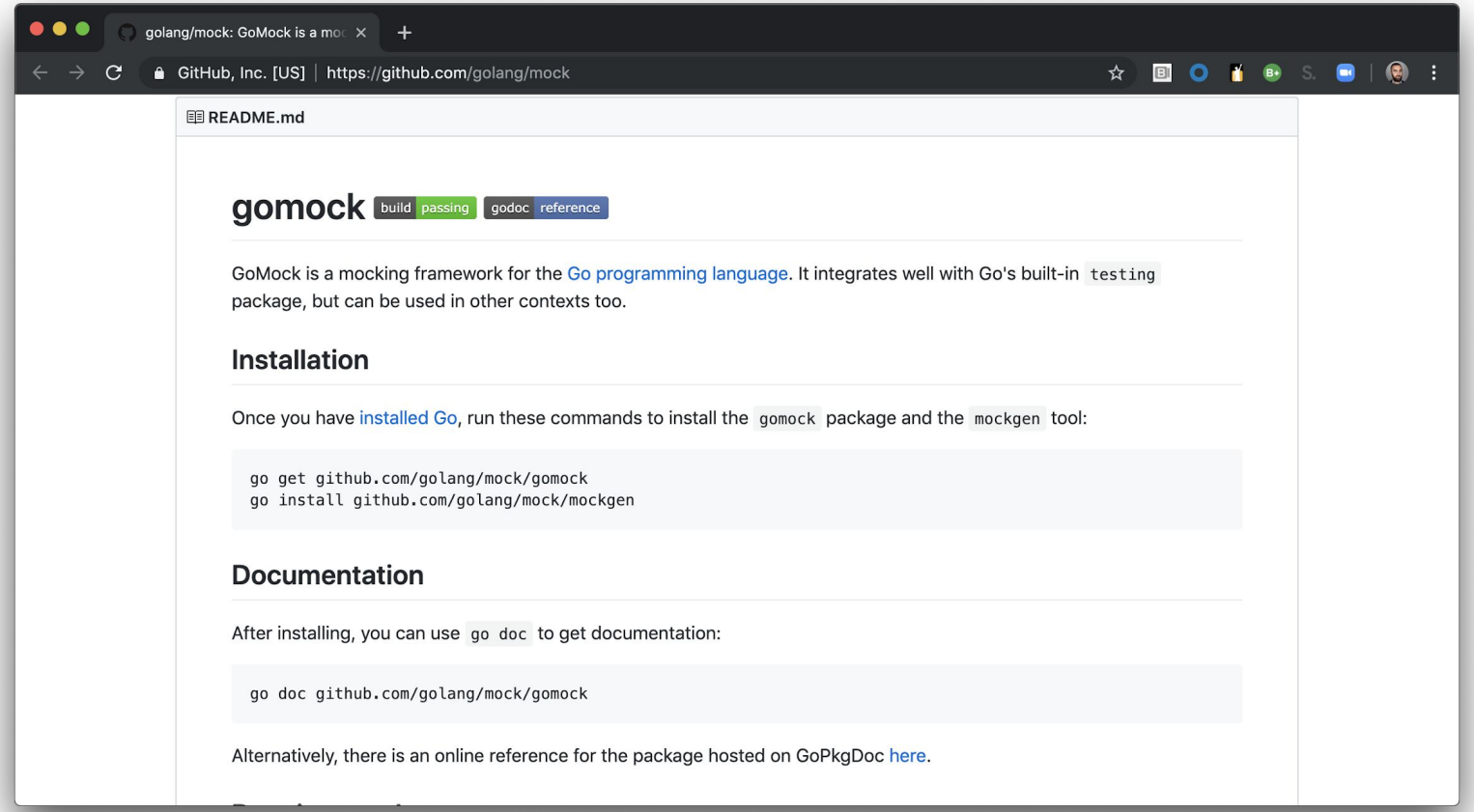
7. **Встроенная поддержка промежуточных обработчиков (middleware).**

   - before/after hooks
   - observability
   - access to metadata

Input

- Interface
- Methods
- Arguments
- Tags

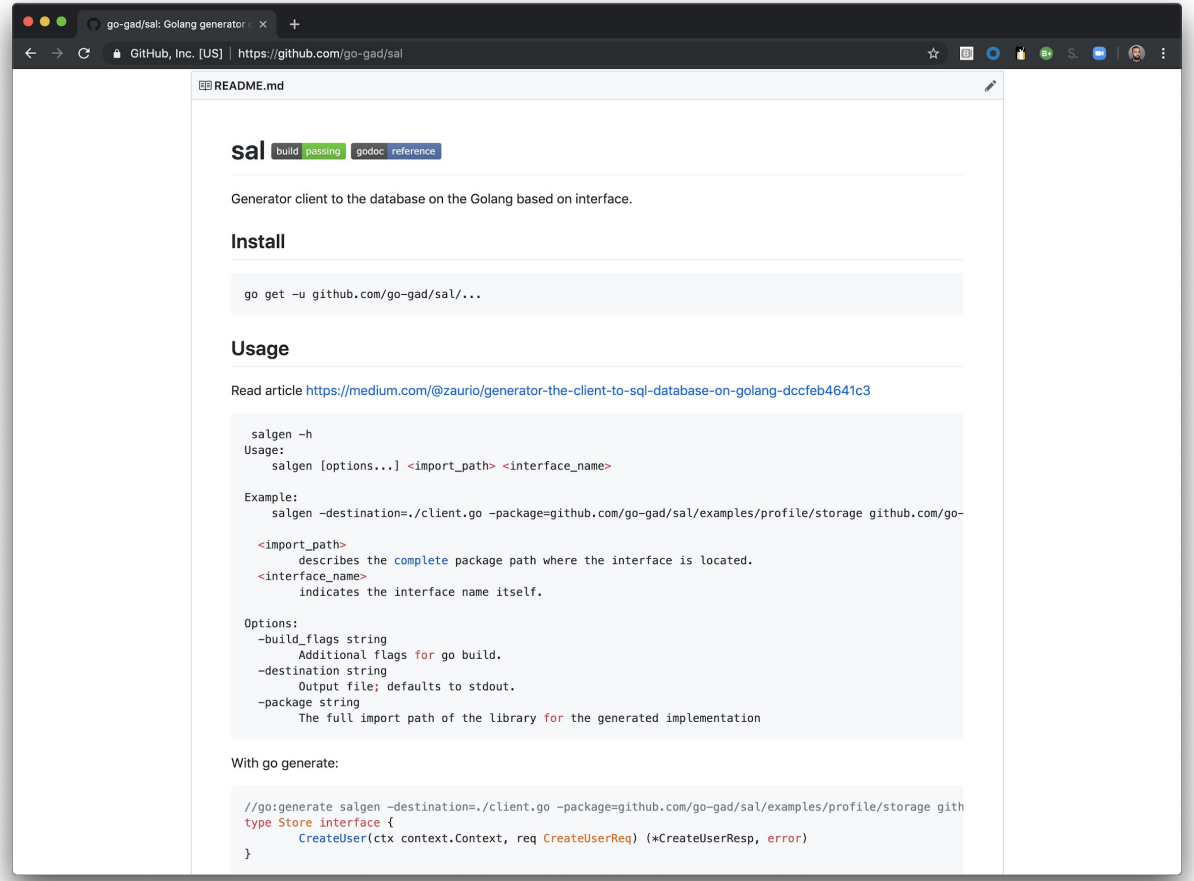
Reference library



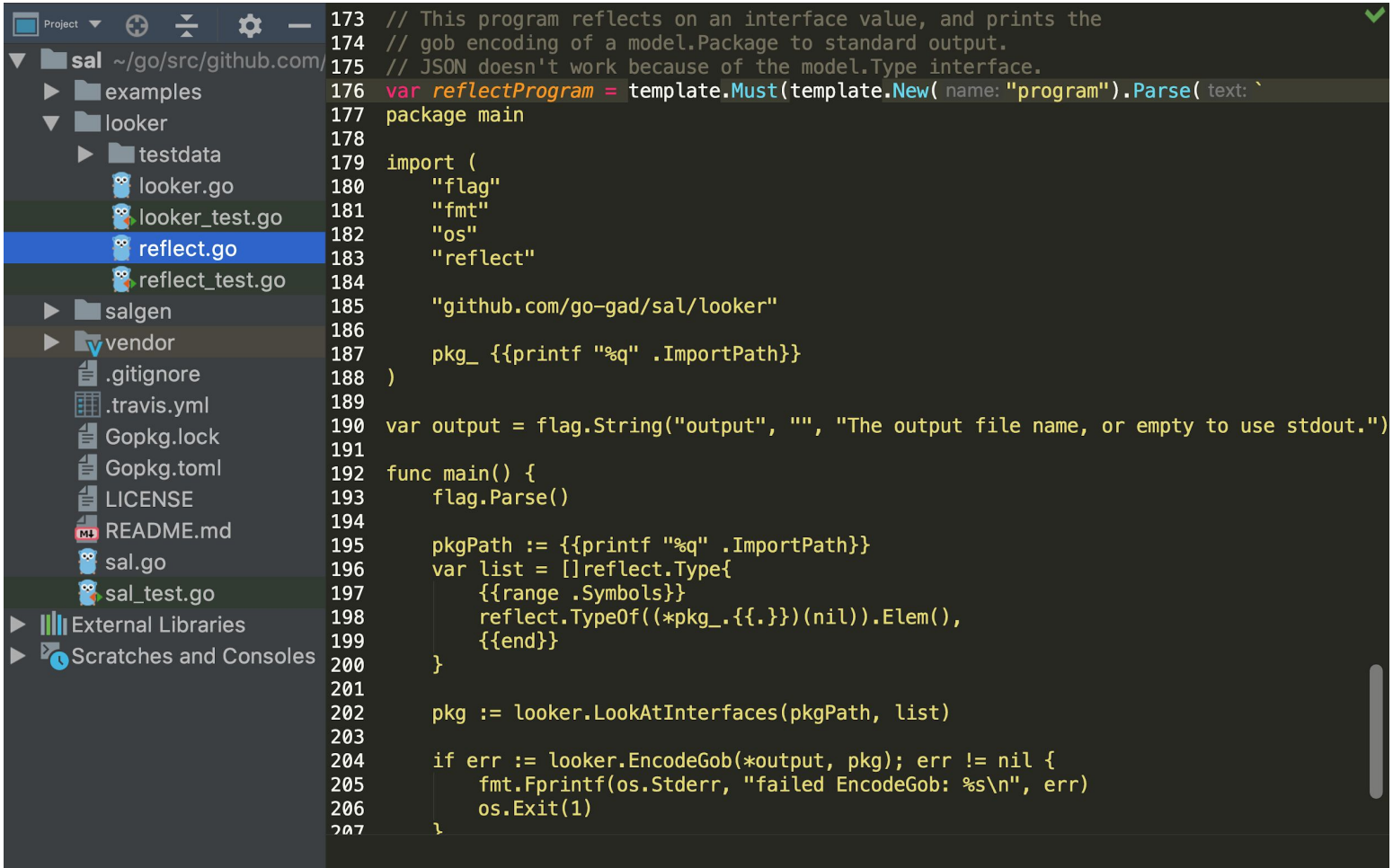
Implementation

1. script input: package name, interfaces, destination file
2. build new script
   a. run temporary script
   b. analyse by reflect
   c. encode (gob)
3. decode
4. generate code of client library
5. write a file

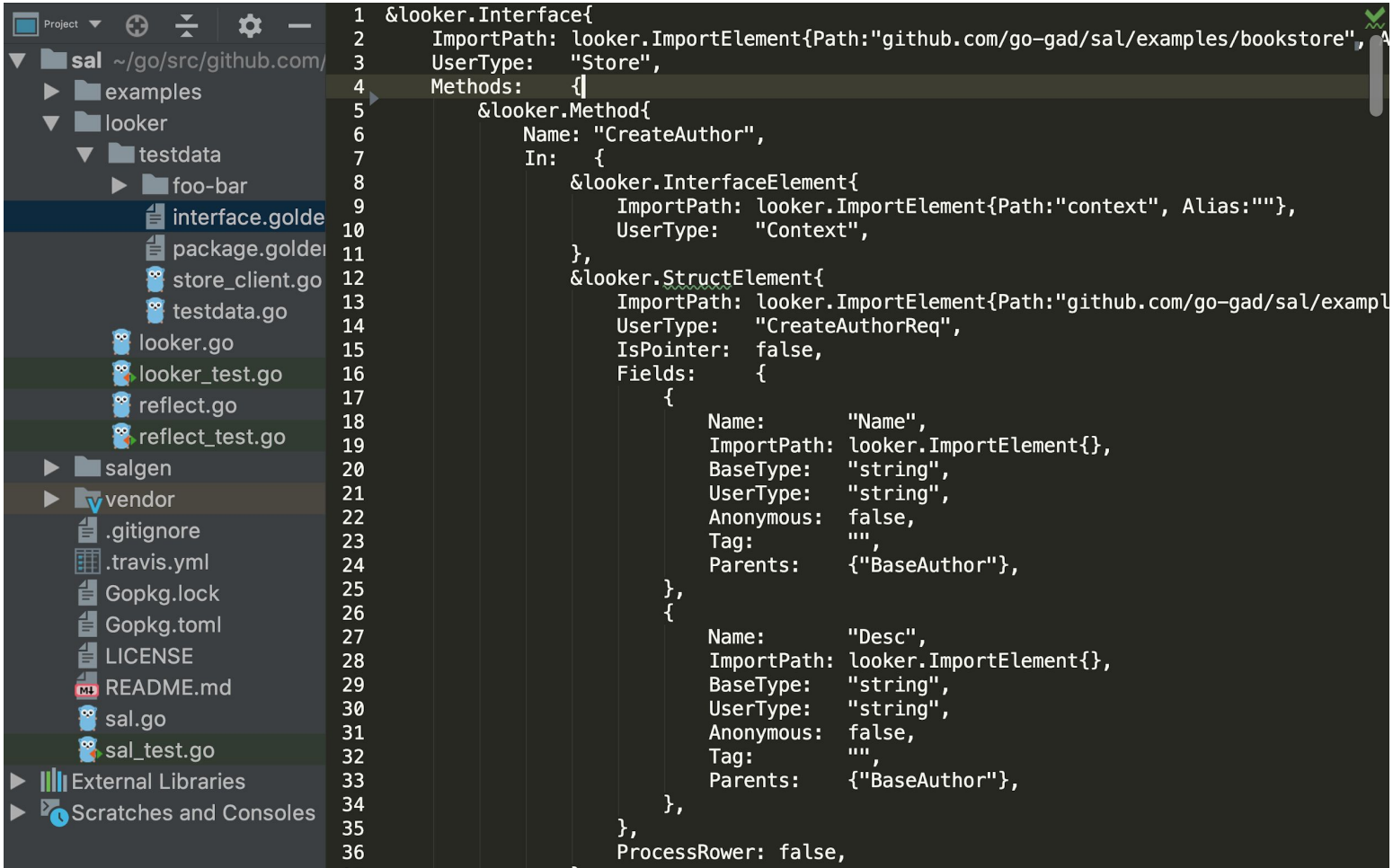
Solution: go-gad/sal



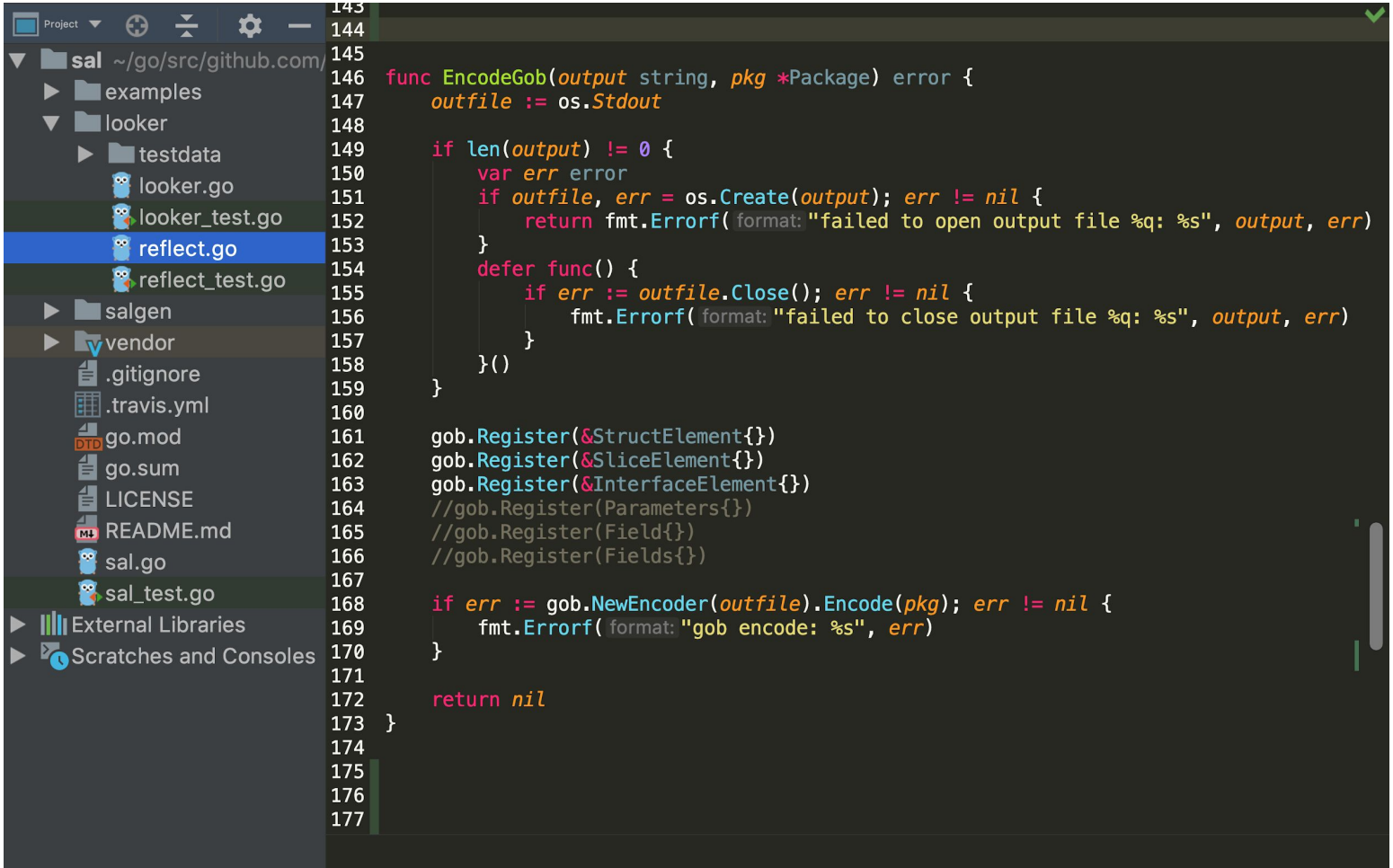
Implementation



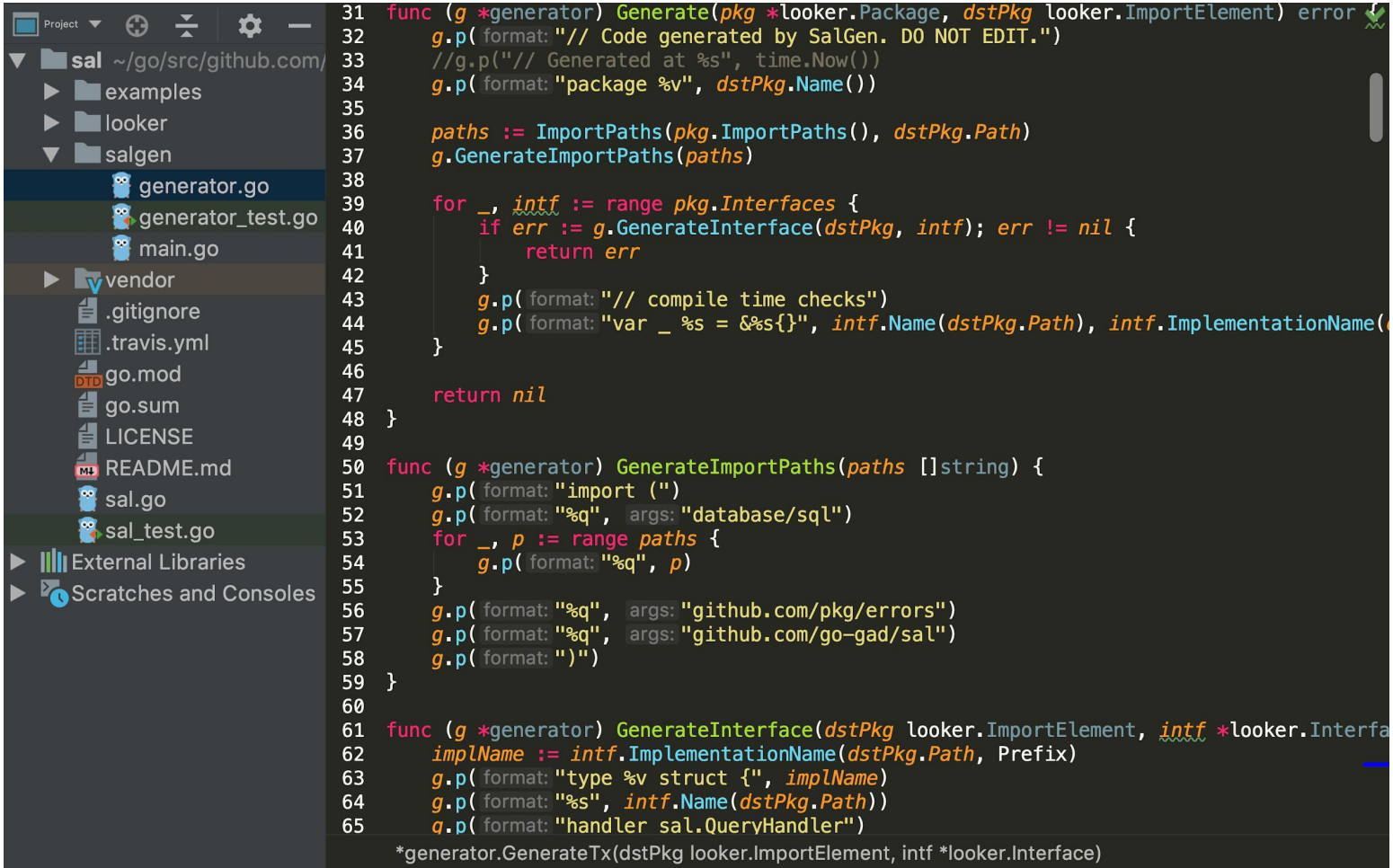
Implementation



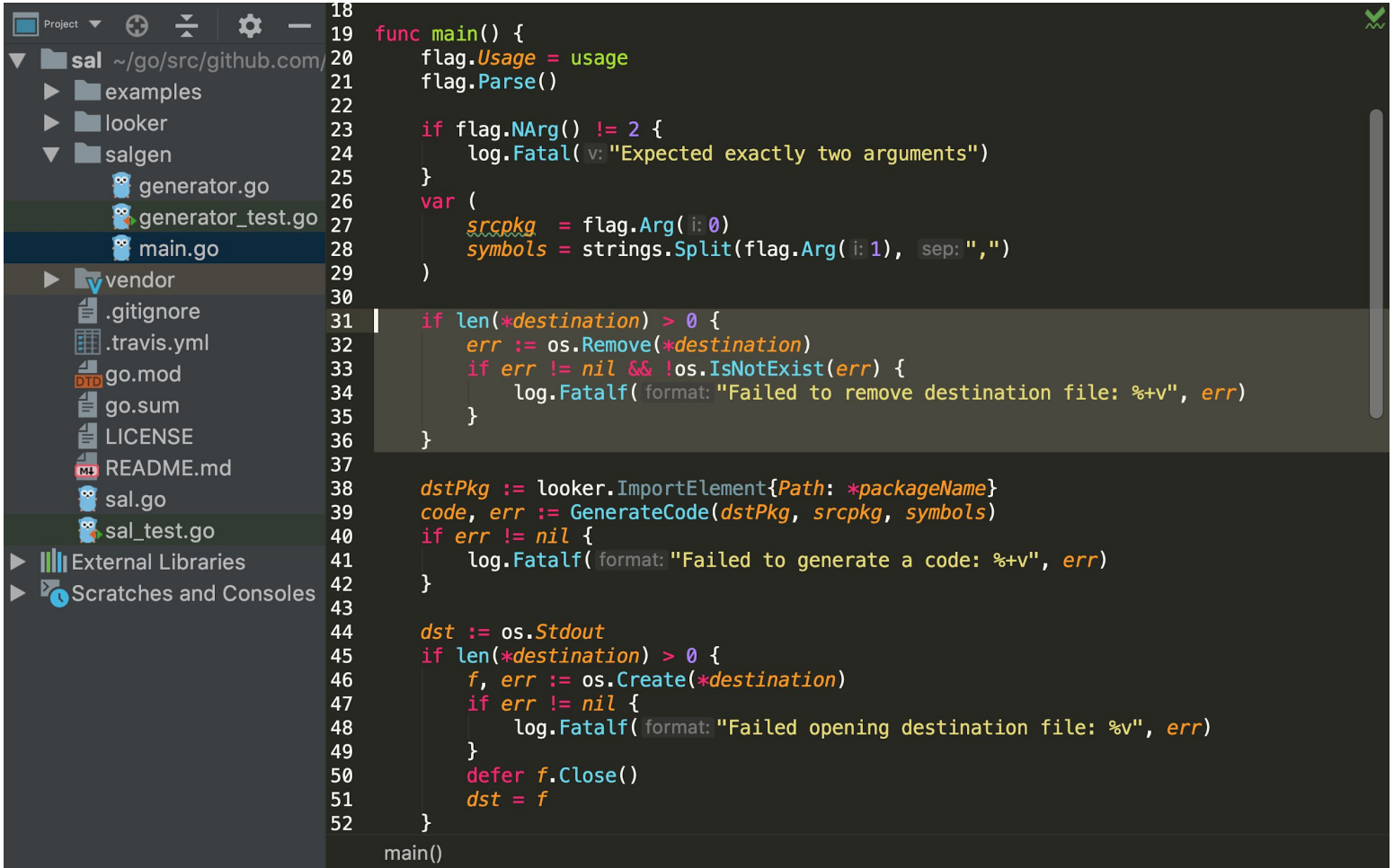
Implementation



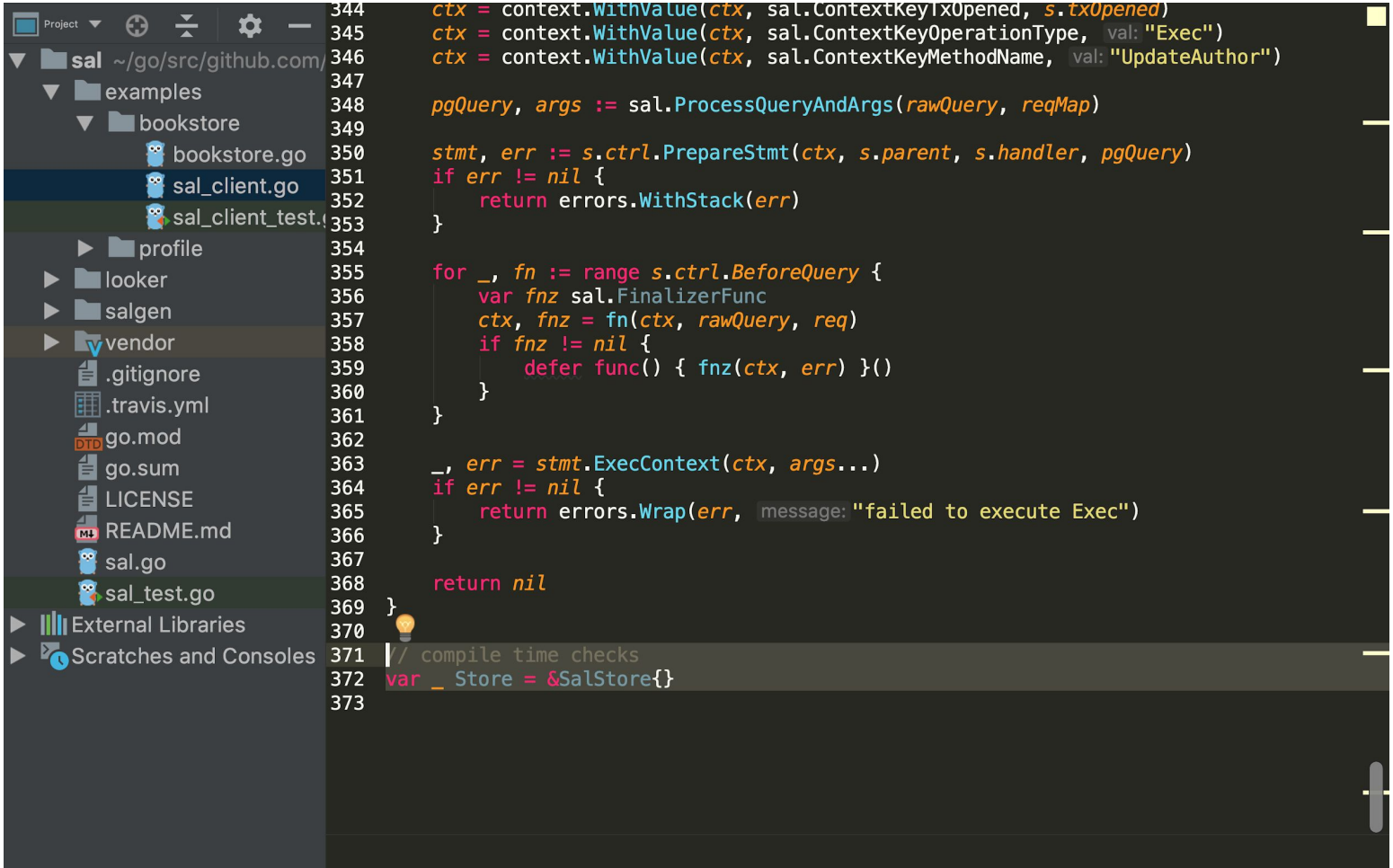
Implementation



Implementation



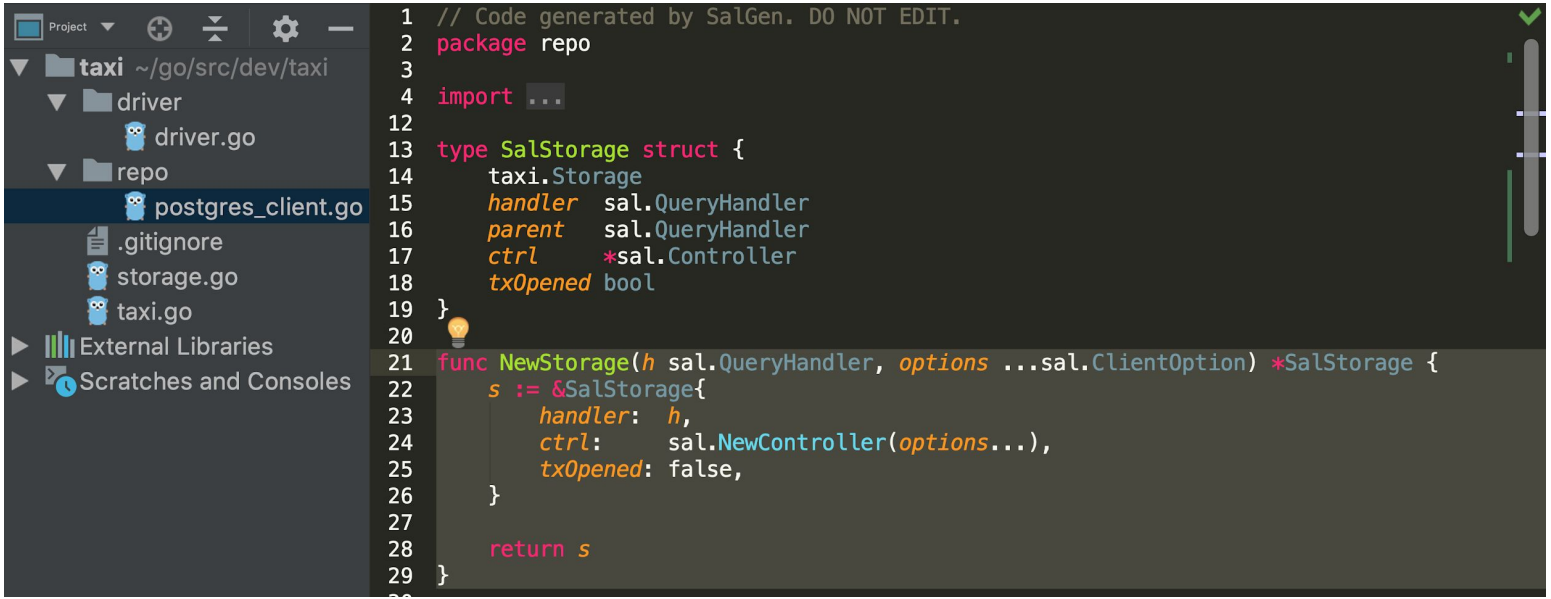
Implementation



Usage: interface

```go
package taxi

import "context"

//go:generate salgen -destination=./repo/postgres_client.go -package=dev/taxi/repo dev/taxi Storage
type Storage interface {
    FindDriverByID(ctx context.Context, req *FindDriverByIDReq) (*Driver, error)
}

type FindDriverByIDReq struct {
    ID string `sql:"id"`
}

func (r *FindDriverByIDReq) Query() string {
    return `"SELECT name FROM drivers WHERE id=@id"`
}
```

Usage: constructor

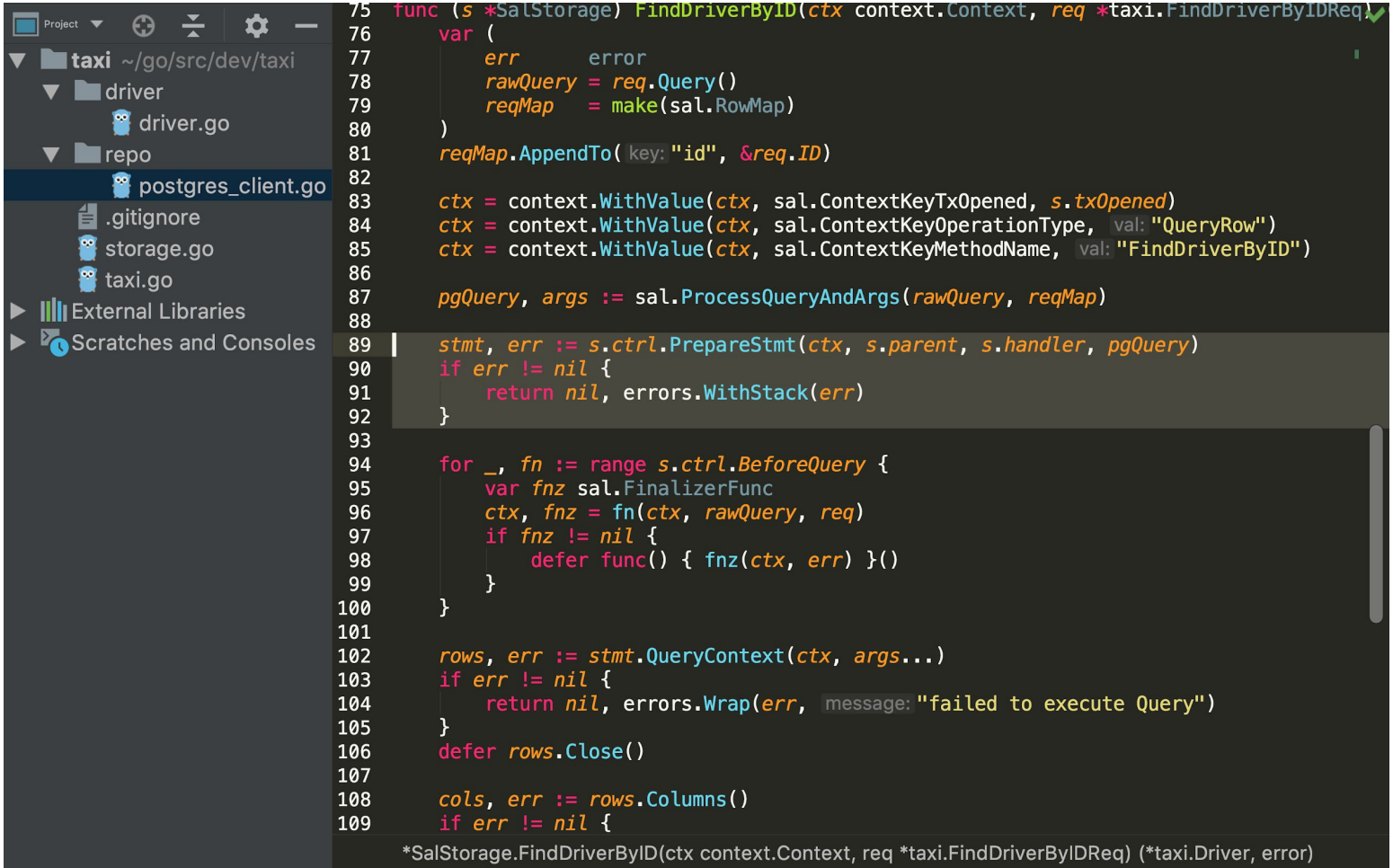



```
db, err := sql.Open("postgres", "postgres://localhost/taxi?sslmode=disable")
store := repo.NewStorage(db)
```

Usage: methods

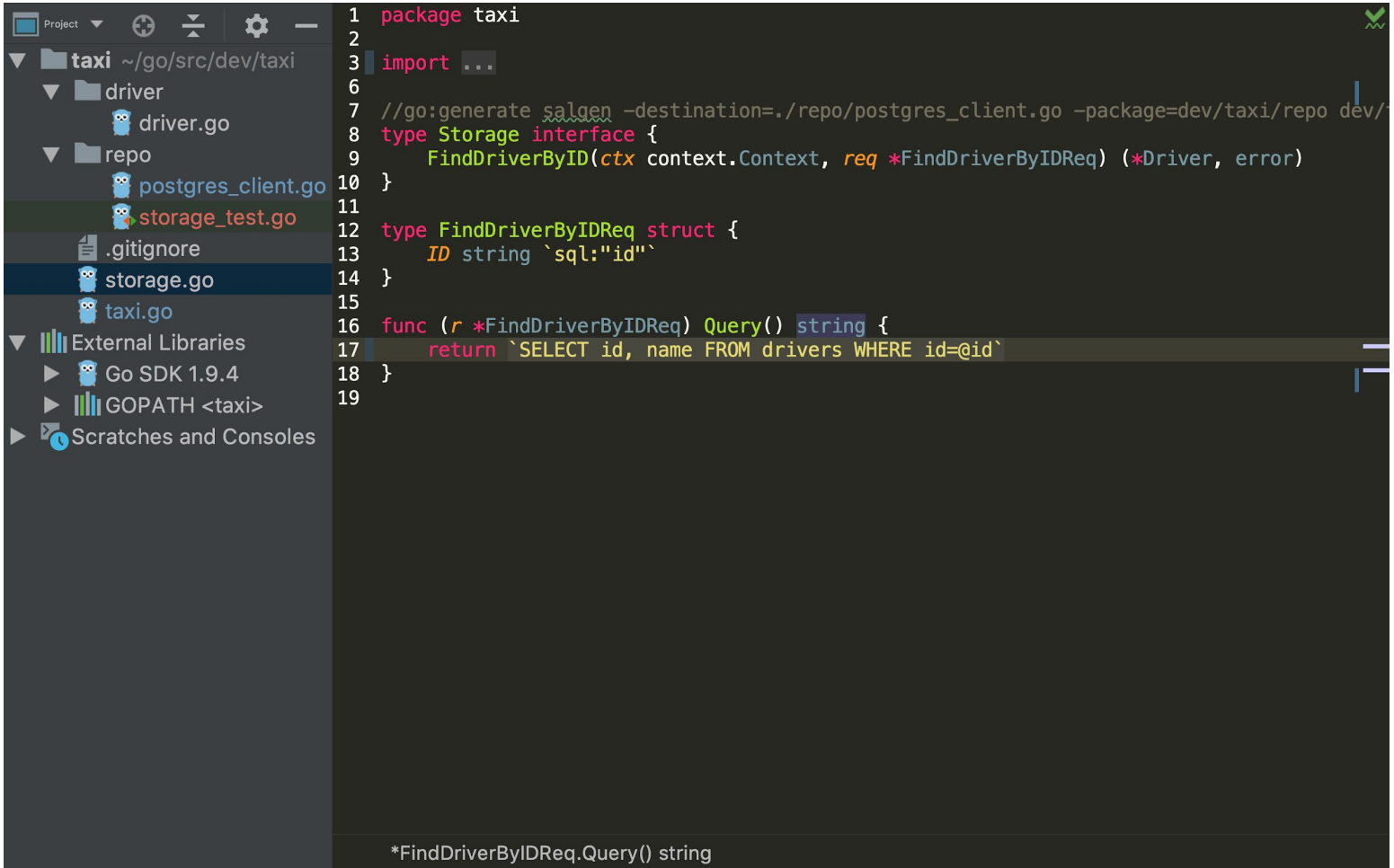
```
type Store interface {
    CreateAuthor(ctx context.Context, req CreateAuthorReq) (CreateAuthorResp, error)
    GetAuthors(ctx context.Context, req GetAuthorsReq) ([]*GetAuthorsResp, error)
    UpdateAuthor(ctx context.Context, req *UpdateAuthorReq) error
}
```

Usage: prepared statements

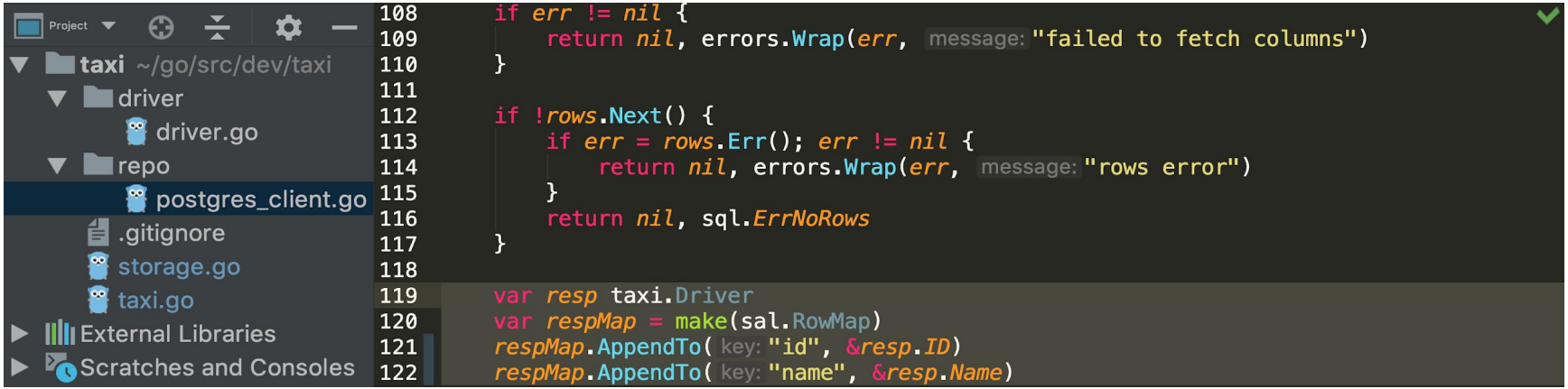



```
75  func (s *SalStorage) FindDriverByID(ctx context.Context, req *taxi.FindDriverByIDReq)
76      var (
77          err      error
78          rawQuery = req.Query()
79          reqMap   = make(sal.RowMap)
80      )
81      reqMap.AppendTo( key: "id", &req.ID)
82
83      ctx = context.WithValue(ctx, sal.ContextKeyTxOpened, s.txOpened)
84      ctx = context.WithValue(ctx, sal.ContextKeyOperationType, val: "QueryRow")
85      ctx = context.WithValue(ctx, sal.ContextKeyMethodName, val: "FindDriverByID")
86
87      pgQuery, args := sal.ProcessQueryAndArgs(rawQuery, reqMap)
88
89      stmt, err := s.ctrl.PrepareStmt(ctx, s.parent, s.handler, pgQuery)
90      if err != nil {
91          return nil, errors.WithStack(err)
92      }
93
94      for _, fn := range s.ctrl.BeforeQuery {
95          var fnz sal.FinalizerFunc
96          ctx, fnz = fn(ctx, rawQuery, req)
97          if fnz != nil {
98              defer func() { fnz(ctx, err) }()
99          }
100     }
101
102     rows, err := stmt.QueryContext(ctx, args...)
103     if err != nil {
104         return nil, errors.Wrap(err, message: "failed to execute Query")
105     }
106     defer rows.Close()
107
108     cols, err := rows.Columns()
109     if err != nil {
```

Usage: named arguments



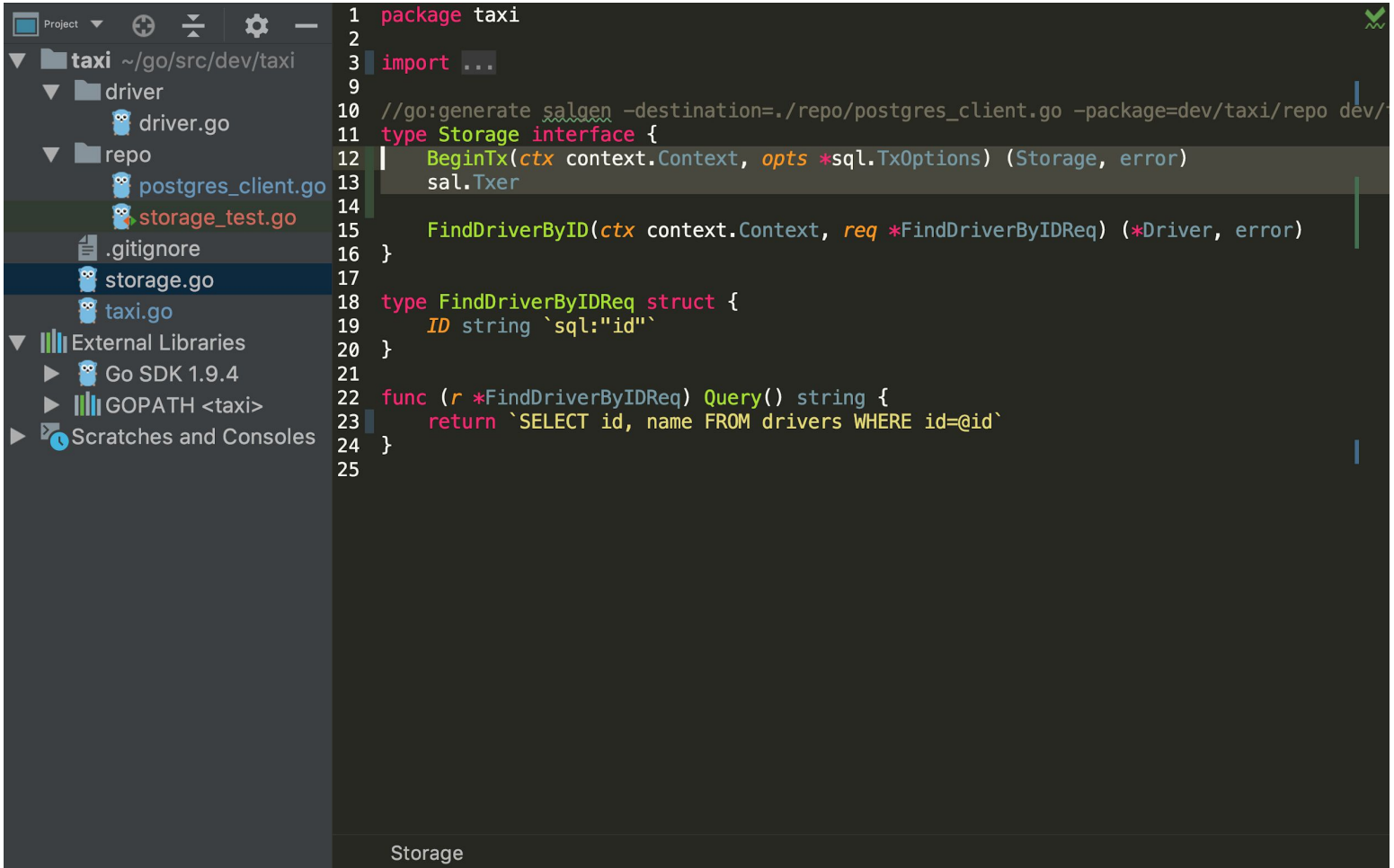
Usage: mapping



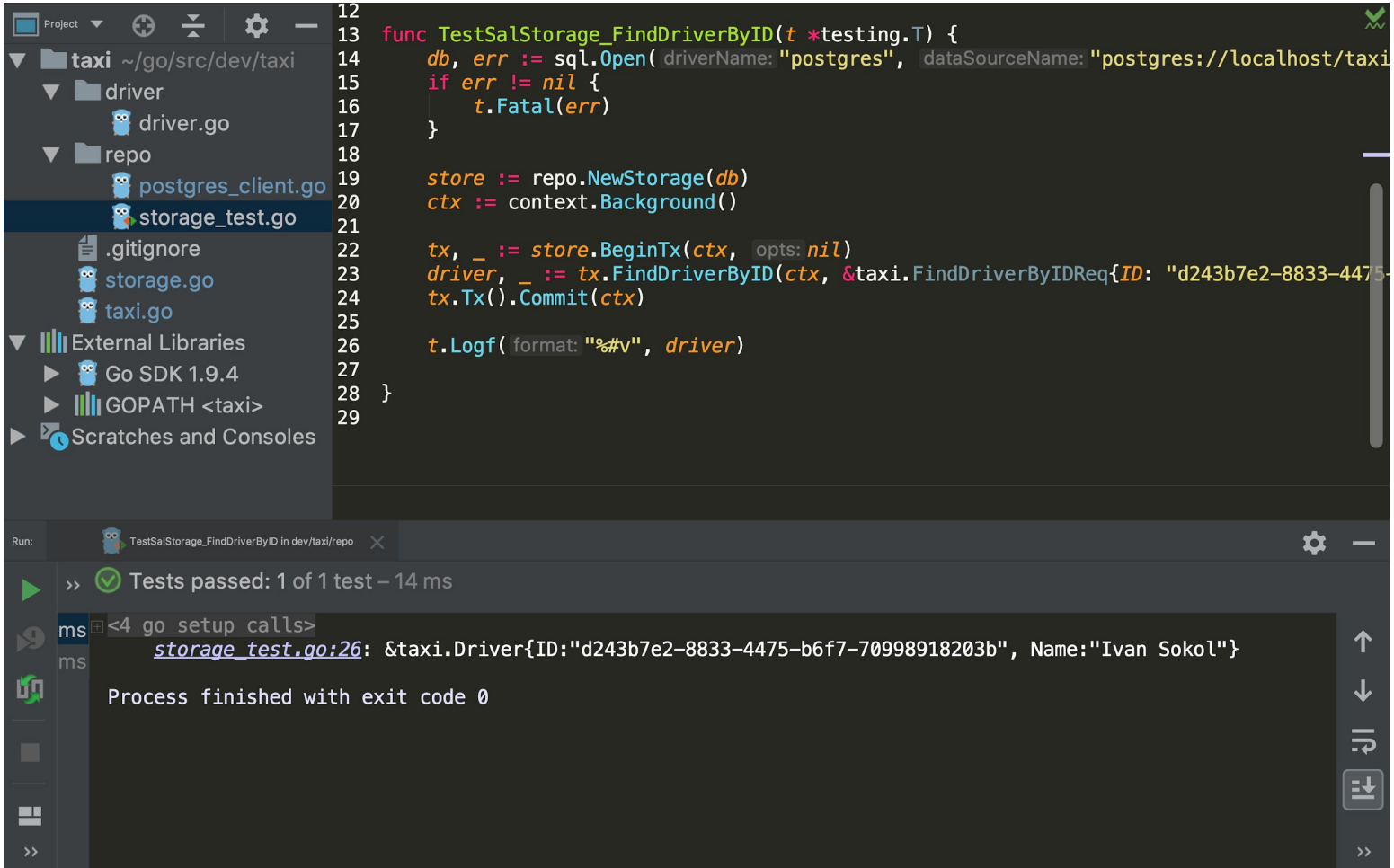
Usage: complex data types

```
type ProcessRower interface {
    ProcessRow(rowMap RowMap)
}

type DeleteAuthrosReq struct {
    Tags []int64 `sql:"tags"`
}

func (r *DeleteAuthorsReq) ProcessRow(rowMap sal.RowMap) {
    rowMap.Set("tags", pq.Array(r.Tags))
}

func (r *DeleteAuthorsReq) Query() string {
    return `DELETE FROM authors WHERE tags=ANY(@tags::UUID[])`
}
```

Usage: transactions



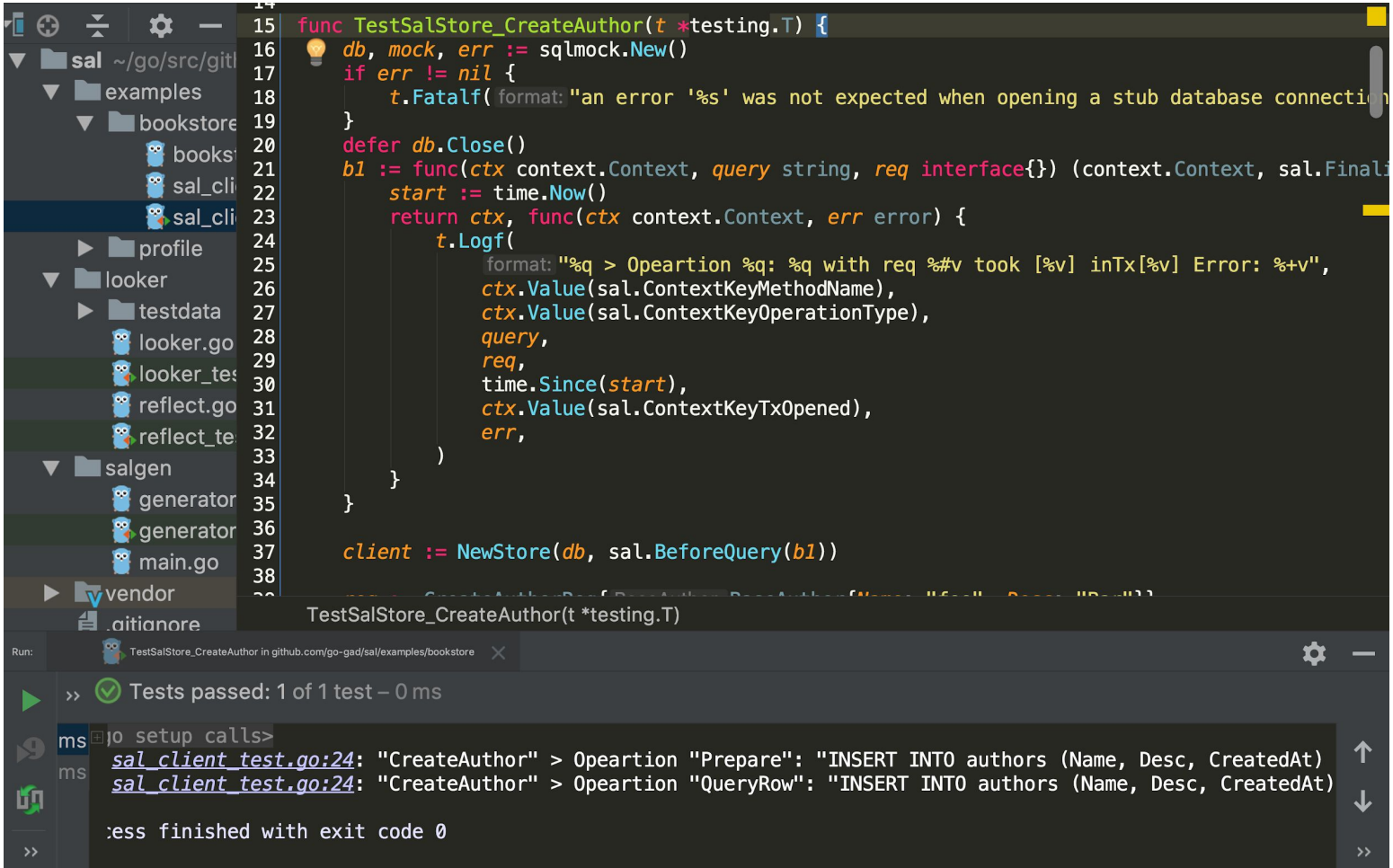
Usage: transactions



Usage: hooks

```
// BeforeQueryFunc is called before the query execution but after the preparing stmts.
// Returns the FinalizerFunc.
type BeforeQueryFunc func(ctx context.Context, query string, req interface{}) (context.Context,
FinalizerFunc)

// FinalizerFunc is executed after the query execution.
type FinalizerFunc func(ctx context.Context, err error)
```

Usage


sal ~/go/src/gitl
examples
bookstore
books
sal_cli
sal_cli
profile
looker
testdata
looker.go
looker_tes
reflect.go
reflect_te
salgen
generator
generator
main.go
vendor
.gitignore
15 func TestSalStore_CreateAuthor(t *testing.T) {
16 db, mock, err := sqlmock.New()
17 if err != nil {
18 t.Fatalf(format: "an error '%s' was not expected when opening a stub database connecti
19 }
20 defer db.Close()
21 b1 := func(ctx context.Context, query string, req interface{}) (context.Context, sal.Finali
22 start := time.Now()
23 return ctx, func(ctx context.Context, err error) {
24 t.Logf(
25 format: "%q > Opeartion %q: %q with req %#v took [%v] inTx[%v] Error: %+v",
26 ctx.Value(sal.ContextKeyMethodName),
27 ctx.Value(sal.ContextKeyOperationType),
28 query,
29 req,
30 time.Since(start),
31 ctx.Value(sal.ContextKeyTxOpened),
32 err,
33)
34 }
35 }
36
37 client := NewStore(db, sal.BeforeQuery(b1))
38
TestSalStore_CreateAuthor(t *testing.T)
Run:
TestSalStore_CreateAuthor in github.com/go-gad/sal/examples/bookstore
Tests passed: 1 of 1 test – 0 ms
o setup calls>
sal_client_test.go:24: "CreateAuthor" > Opeartion "Prepare": "INSERT INTO authors (Name, Desc, CreatedAt)
sal_client_test.go:24: "CreateAuthor" > Opeartion "QueryRow": "INSERT INTO authors (Name, Desc, CreatedAt)
ess finished with exit code 0

Features list

- Generated client based on the interface
- Prepared statements
- Named arguments
- Mapping
- Transactions
- Middleware

https://godoc.org/github.com/go-gad/sal

Thank you
Zaur Abasmirzoev
Gett
http://zaur.io