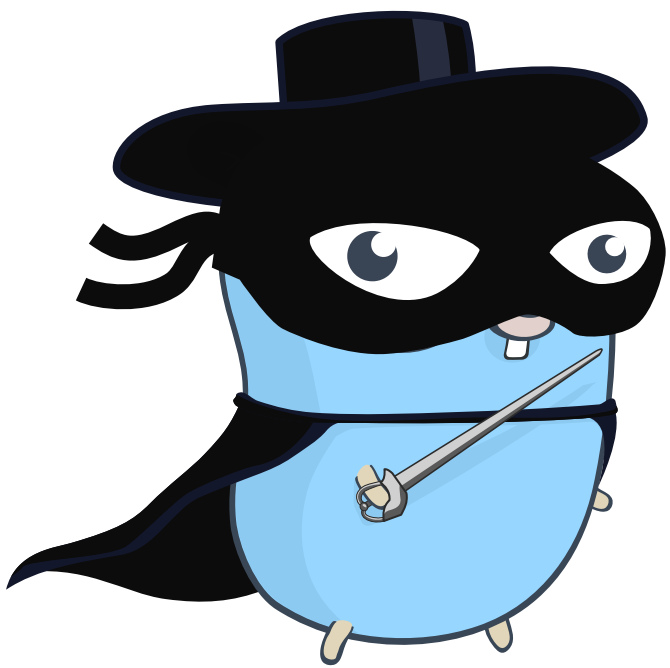
Generating Code in Go

Different approach to some challenges

(no AI)



Gopher: https://github.com/egonelbre/gophers

Who am I?

Daniel Antos

I *used to* hate generated code.

Staff Backend Engineer at Pento

pento.io

What are we talking about today?

- Why?
- Basics of generating code in Go.
- Generate your APIs: gRPC, GraphQL.
- Generate your ORM.
- Generating decorators.
- Everyday life with generated code.

Why?

- Compile time safety
- To reduce abstraction and complexity
- Computer is better at repetitive task

```
Error while building:
 # github.com/antosdaniel/go-presentation-generate-code/internal/grpc
internal/grpc/payrollServiceWithAuth.go:16:9: ... (missing method GetPayslip)
```

Basics of generating code in Go

Generators are just CLI tools:

```
.PHONY: generate
generate:
    @printf "Generating protos...\n"
    @buf generate --template gen/grpc/buf.gen.yaml
```

```
//go:generate gowrap gen -g
```

go generate <path>

go generate ./...

Generate your APIs: gRPC, GraphQL

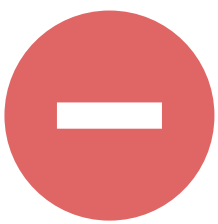
```
1  syntax = "proto3";
2
3  package payroll.v1;
4
5  service PayrollService {
6    rpc AddPayroll(AddPayrollRequest) returns (AddPayrollResponse);
7    rpc AddPayslip(AddPayslipRequest) returns (AddPayslipResponse);
8    rpc GetPayroll(GetPayrollRequest) returns (GetPayrollResponse);
9  }
10
11 message AddPayrollRequest {
12   string payroll_id = 1;
13   string tenant_id = 2;
14   Date   payday = 3;
15 }
16
17 // ...
```

Generate your APIs: gRPC, GraphQL, REST

```
1  // PayrollServiceHandler is an implementation of the payroll.v1.PayrollServic
2  type PayrollServiceHandler interface {
3    AddPayroll(context.Context, *connect_go.Request[payrollv1.AddPayrollReques
4    AddPayslip(context.Context, *connect_go.Request[payrollv1.AddPayslipReques
5    GetPayroll(context.Context, *connect_go.Request[payrollv1.GetPayrollReques
6  }
7
8  // Now we only have to implement it:
9  func (s *payrollServiceServer) AddPayroll(
10   ctx context.Context,
11   request *connect_go.Request[payrollv1.AddPayrollRequest],
12 ) (*connect_go.Response[payrollv1.AddPayrollResponse], error) {
13   // TODO: business logic
14
15   return &connect_go.Response[payrollv1.AddPayrollResponse]{
16     Msg: &payrollv1.AddPayrollResponse{ PayrollId: id, },
17   }, nil
18 }
```

Generate your ORM

```
create table payrolls (
    id        uuid not null primary key,
    tenant_id uuid not null,
    payday    date not null
);
```

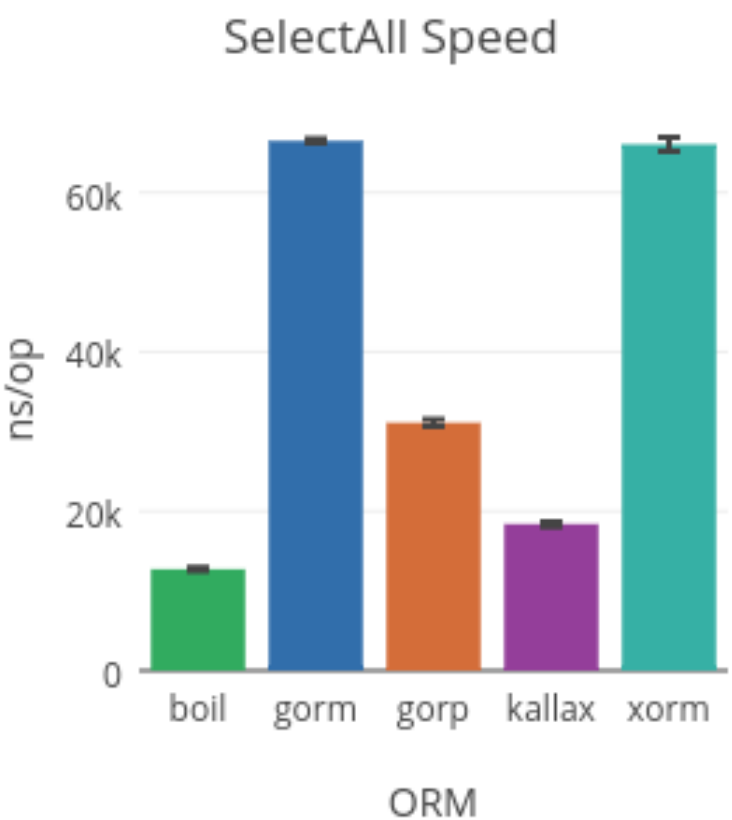


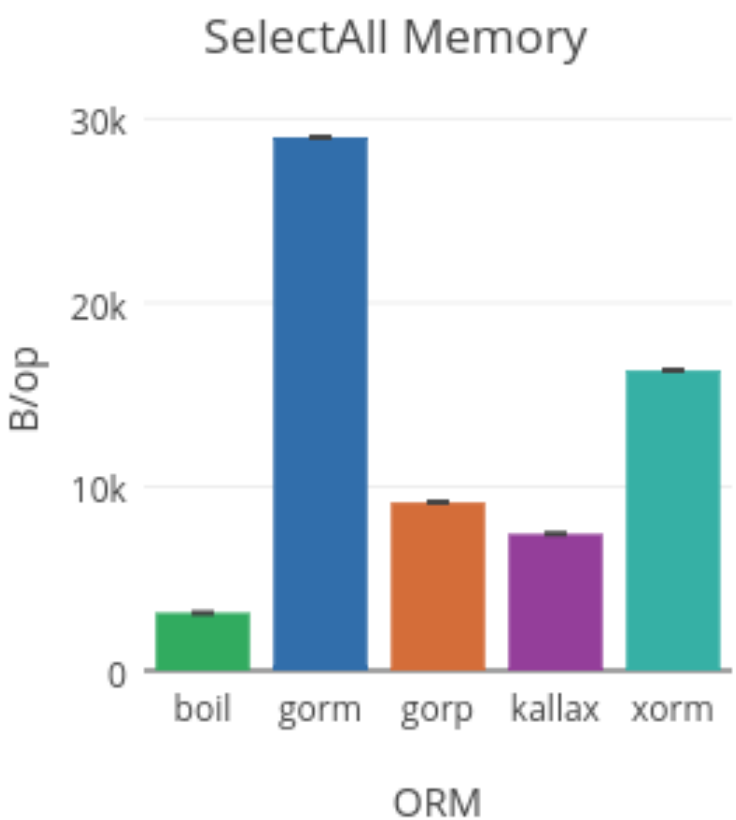
```
type PayrollModel struct {
        ID       string `orm:"id"`
        TenantID string `orm:"tenant_id"`
        Payday   string `orm:"payday"`
}
```

The ugly side of abstraction

```
1  func FindPayroll(ctx context.Context, exec boil.ContextExecutor, iD string, s
2          payrollObj := &Payroll{}
3
4          sel := "*"
5          if len(selectCols) > 0 {
6                  sel = strings.Join(strmangle.IdentQuoteSlice(dialect.LQ, dial
7          }
8          query := fmt.Sprintf(
9                  "select %s from \"payrolls\" where \"id\"=$1", sel,
10         )
11
12         q := queries.Raw(query, iD)
13
14         err := q.Bind(ctx, exec, payrollObj)
15         if err != nil {
16                 if errors.Is(err, sql.ErrNoRows) {
17                         return nil, sql.ErrNoRows
18                 }
19                 return nil, errors.Wrap(err, "models: unable to select from p
20         }
21
22         return payrollObj, nil
```

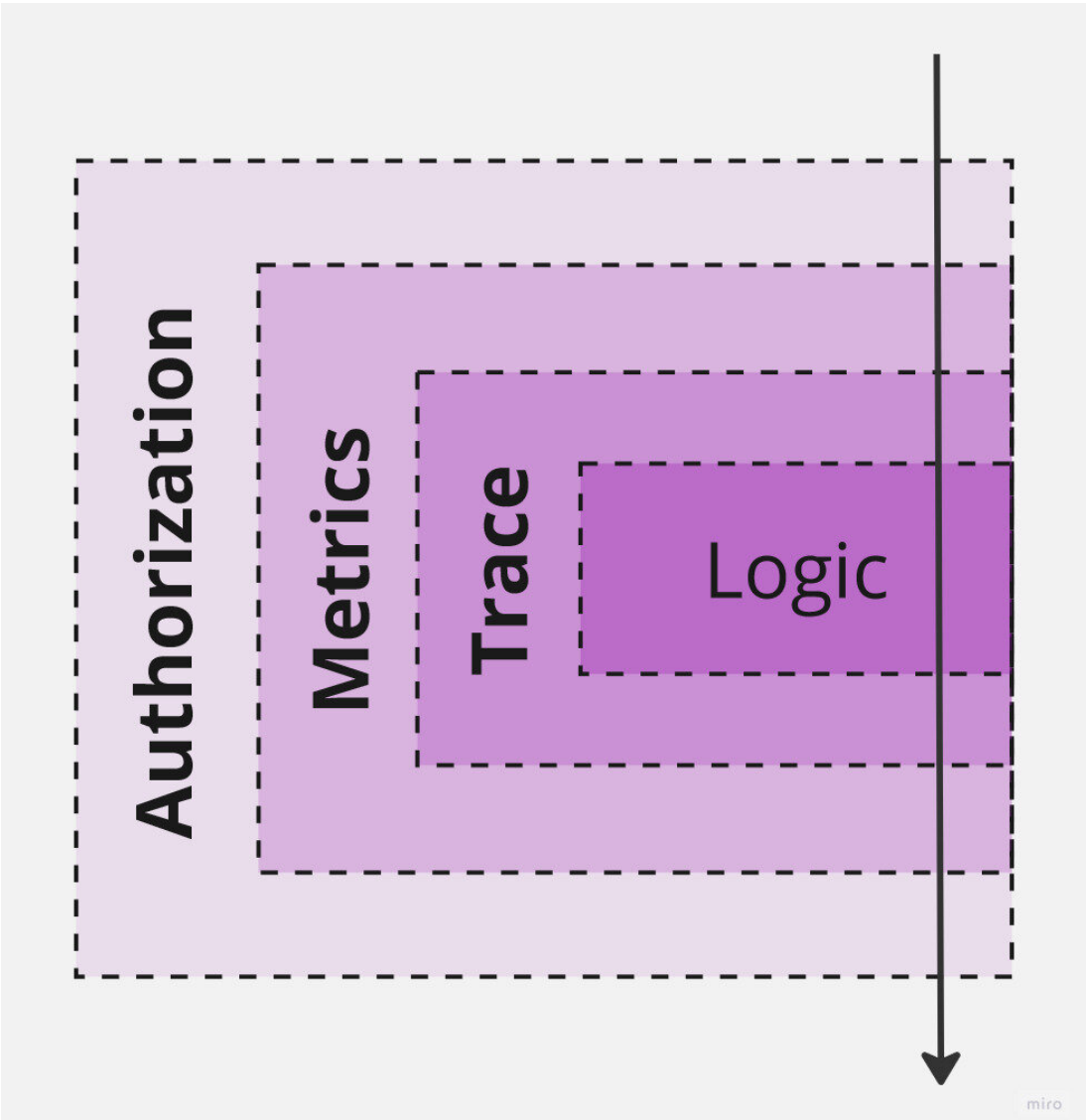
Abstraction and reflection are costly





Source: https://github.com/volatiletech/sqlboiler

Generating decorators



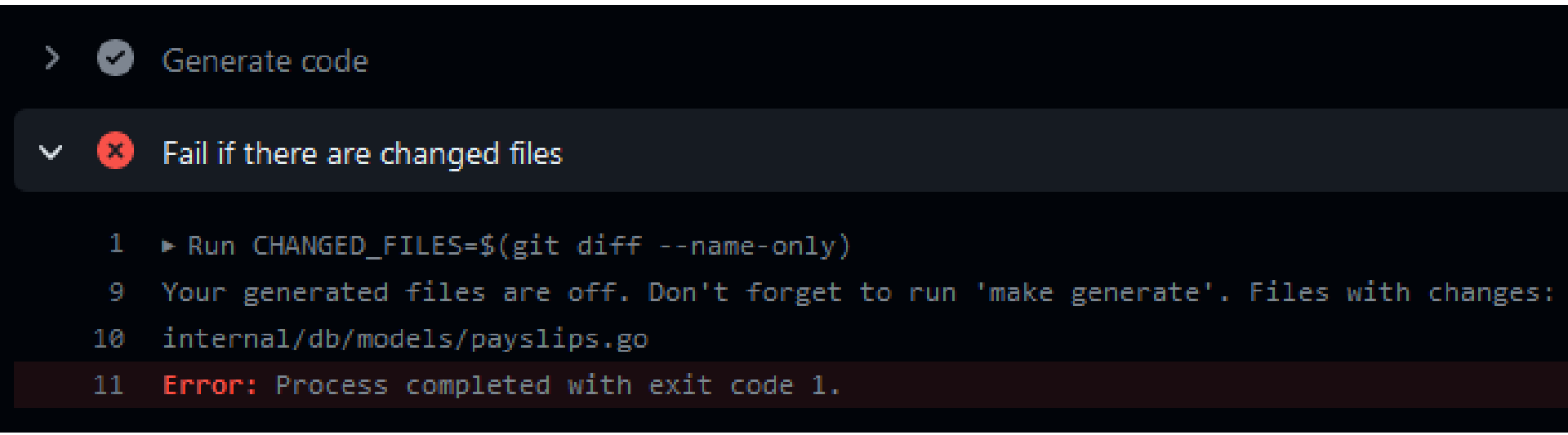
What can we generate?

```
1  func (ps PayrollServiceWithTrace) AddPayroll(...) error {
2          ps.tracer.Start(...)
3          defer ps.tracer.End(...)
4
5          err := ps.base.AddPayroll(...)
6
7          if err != nil {
8                  ps.tracer.RaiseError(err)
9          }
10         return err
11 }
12
13 func (ps PayrollServiceWithTrace) AddPayslip(...) error {
14         ps.tracer.Start(...)
15         defer ps.tracer.End(...)
16
17         err := ps.base.AddPayslip(...)
18
19         if err != nil {
20                 ps.tracer.RaiseError(err)
21         }
22         return err
23 }
```

Everyday life with generated code

because you will hit some issues

Commit generated code, and check it in CI



- No need to generate code before build
- Easy to spot unintended changes during code review
- Easily reproducible builds

Makefile is where it begins

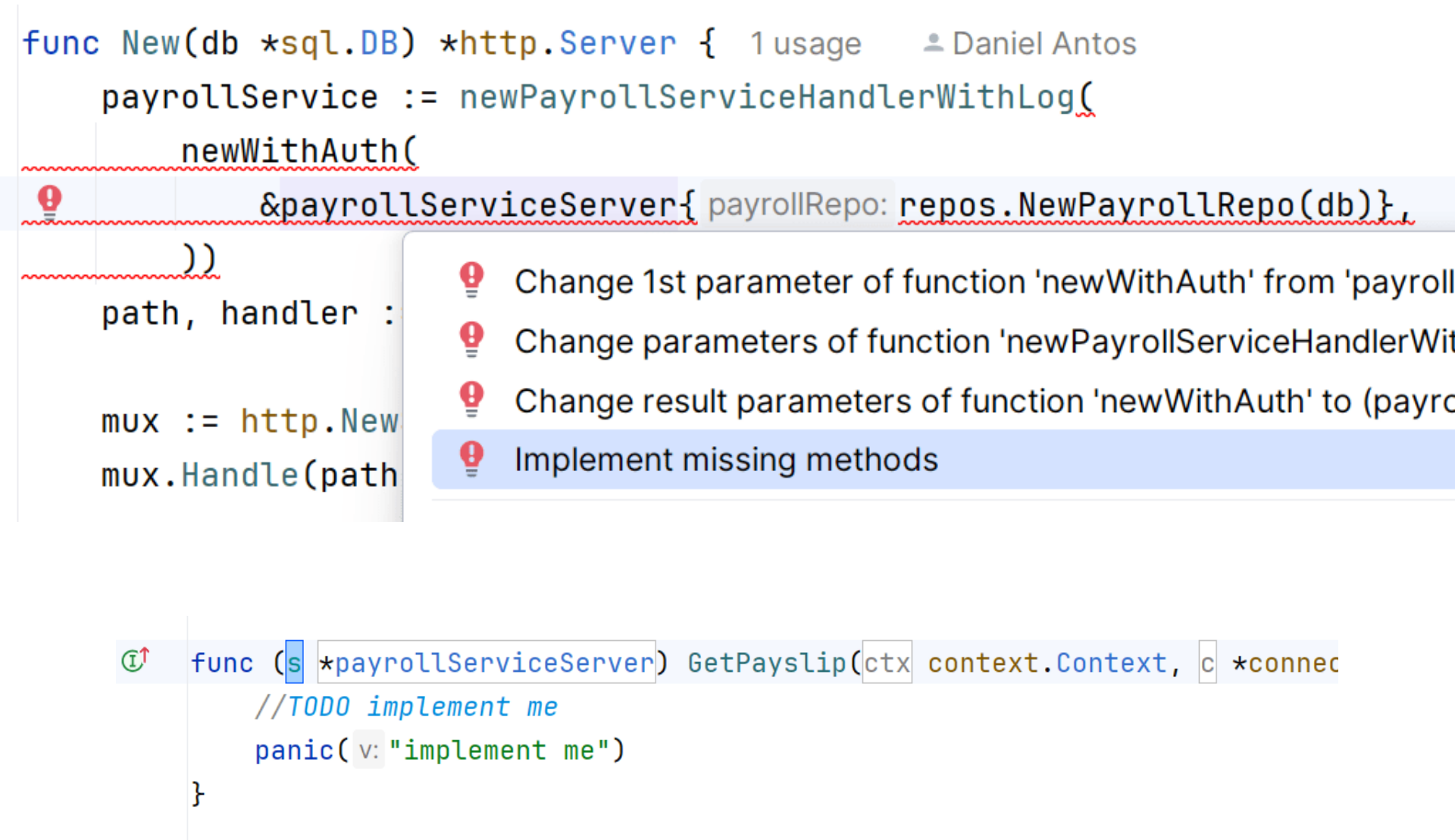
```
1  .PHONY: install
2  install:
3    @printf "\nInstalling sqlboiler...\n"
4    @go install -mod=readonly github.com/volatiletech/sqlboiler/v4@v4.15.0
5    @go install -mod=readonly github.com/volatiletech/sqlboiler/v4/drivers/sql
6
7    @printf "\nInstalling gowrap...\n"
8    @go install -mod=readonly github.com/hexdigest/gowrap/cmd/gowrap@v1.3.2
9
10   # ...
11
12 .PHONY: generate
13 generate:
14   @printf "Generating protos...\n"
15   @buf generate --template gen/grpc/buf.gen.yaml
16
17   @printf "Generating db models...\n"
18   @sqlboiler --config db/sqlboiler.toml psql
19
20   @printf "go generate...\n"
21   @go generate ./...
22
23   @$(MAKE) format
```

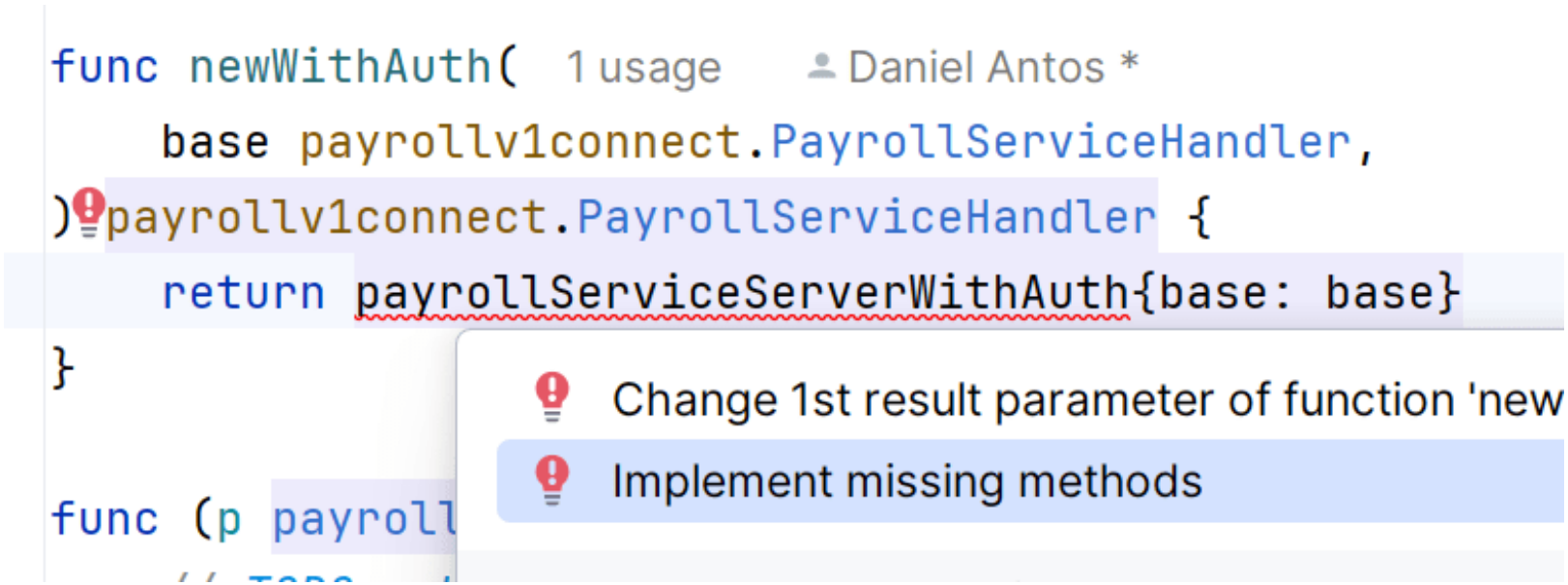
Live coding!

```
9   service PayrollService {
10    rpc AddPayroll(AddPayrollRequest) returns (AddPayrollResponse);
11    rpc AddPayslip(AddPayslipRequest) returns (AddPayslipResponse);
12    rpc GetPayroll(GetPayrollRequest) returns (GetPayrollResponse);
13    rpc GetPayslip(GetPayslipRequest) returns (Payslip);
14  }
15
16  message GetPayslipRequest {
17    string payslip_id = 1;
18  }
```

```
go-presentation-generate-code on  main via  v1.21.0
❯ make generate
Generating protos...
Generating db models...
go generate...
Refreshing Go modules...
```

```
go-grpc-1     | Error while building:
go-grpc-1     |  # github.com/antosdaniel/go-presentation-generate-code
/internal/grpc
go-grpc-1     | internal/grpc/payrollServiceWithLogs.go:26:96: undefine
d: v1
go-grpc-1     | internal/grpc/payrollServiceWithLogs.go:41:96: undefine
d: v1
go-grpc-1     | internal/grpc/payrollServiceWithLogs.go:56:96: undefine
d: v1
go-grpc-1     | internal/grpc/payrollServiceWithLogs.go:71:96: undefine
d: v1
go-grpc-1     | internal/grpc/payrollServiceWithAuth.go:16:9: cannot us
e payrollServiceServerWithAuth{…} (value of type payrollServiceServerWi
thAuth) as payrollv1connect.PayrollServiceHandler value in return state
ment: payrollServiceServerWithAuth does not implement payrollv1connect.
PayrollServiceHandler (missing method GetPayslip)
go-grpc-1     | internal/grpc/setup.go:24:4: cannot use &payrollService
Server{…} (value of type *payrollServiceServer) as payrollv1connect.Pay
rollServiceHandler value in argument to newWithAuth: *payrollServiceSer
ver does not implement payrollv1connect.PayrollServiceHandler (missing
```


func New(db *sql.DB) *http.Server {
1 usage
Daniel Antos
payrollService := newPayrollServiceHandlerWithLog(
newWithAuth(
&payrollServiceServer{ payrollRepo: repos.NewPayrollRepo(db)},
))
path, handler :
mux := http.New
mux.Handle(path
Change 1st parameter of function 'newWithAuth' from 'payroll
Change parameters of function 'newPayrollServiceHandlerWit
Change result parameters of function 'newWithAuth' to (payro
Implement missing methods
func (s *payrollServiceServer) GetPayslip(ctx context.Context, c *connec
//TODO implement me
panic(v: "implement me")
}




```
20
21  func (p payrollServiceServerWithAuth) GetPayslip(ctx
22      // TODO: check permission
23      return p.base.GetPayslip(ctx, c)
24  }
25
```

```
go-grpc-1      | Running build command!
go-grpc-1      | Build ok.
go-grpc-1      | Restarting the given command.
go-grpc-1      | 2023/11/14 20:56:14 starting server...
```

```
go-grpc-1      | 2023/11/14 21:03:45 PayrollServiceHandlerWithLog: calli
ng GetPayslip with params: context.Background.WithValue(type *http.cont
extKey, val <not Stringer>).WithValue(type *http.contextKey, val 172.19
.0.4:8000).WithCancel.WithCancel.WithCancel.WithValue(type *http.contex
tKey, val 172.19.0.4:8000).WithValue(type *http.contextKey, val <not St
ringer>).WithCancel &{payslip_id:"acb374e9-5507-4070-a70e-133324e3f64f"
 {0 /payroll.v1.PayrollService/GetPayslip false} {172.19.0.1:36438 grpc
} map[Accept-Encoding:[identity] Content-Type:[application/grpc] Grpc-A
ccept-Encoding:[identity,deflate,gzip] Te:[trailers] User-Agent:[grpc-n
ode-js/1.8.10]]}
go-grpc-1      | 2023/11/14 21:03:45 PayrollServiceHandlerWithLog: GetPa
yslip returned results: &{id:"acb374e9-5507-4070-a70e-133324e3f64f" map
[] map[]} <nil>
```

What did we learn?

- When you have schema, there is no reason to repeat yourself (DRY)
- Compiler can find mistakes for you
- You can drive improvement through automation

Recommended tools

- gRPC: Buf
- GraphQL: gqlgen
- Database: sqlboiler
- Decorators: gowrap

Thanks!

Any questions for me?

https://github.com/antosdaniel/go-presentation-generate-code