Nubes: Object-Oriented Programming for Stateful Serverless Functions

Kinga Marek, Luca De Martini, Alessandro Margara

POLITECNICO
MILANO 1863

Function as a Service

AWS Lambda

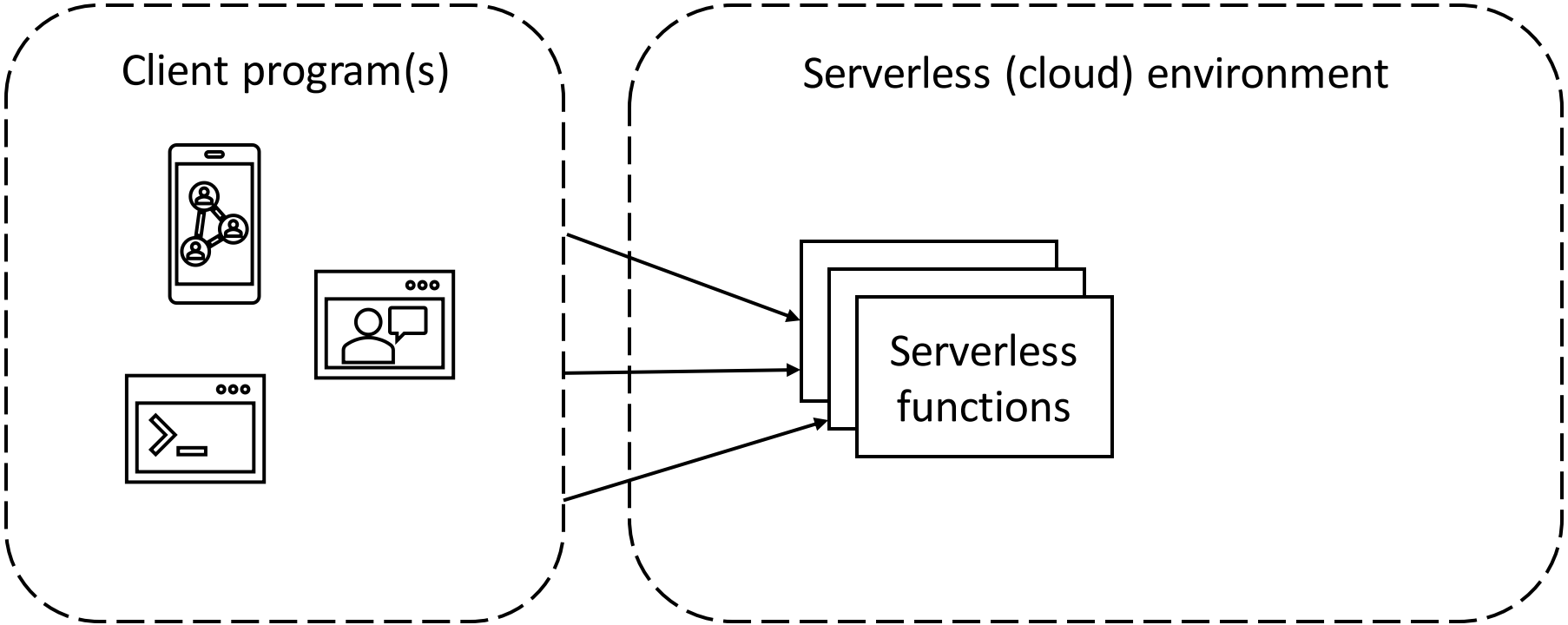
Azure function

Google Cloud function

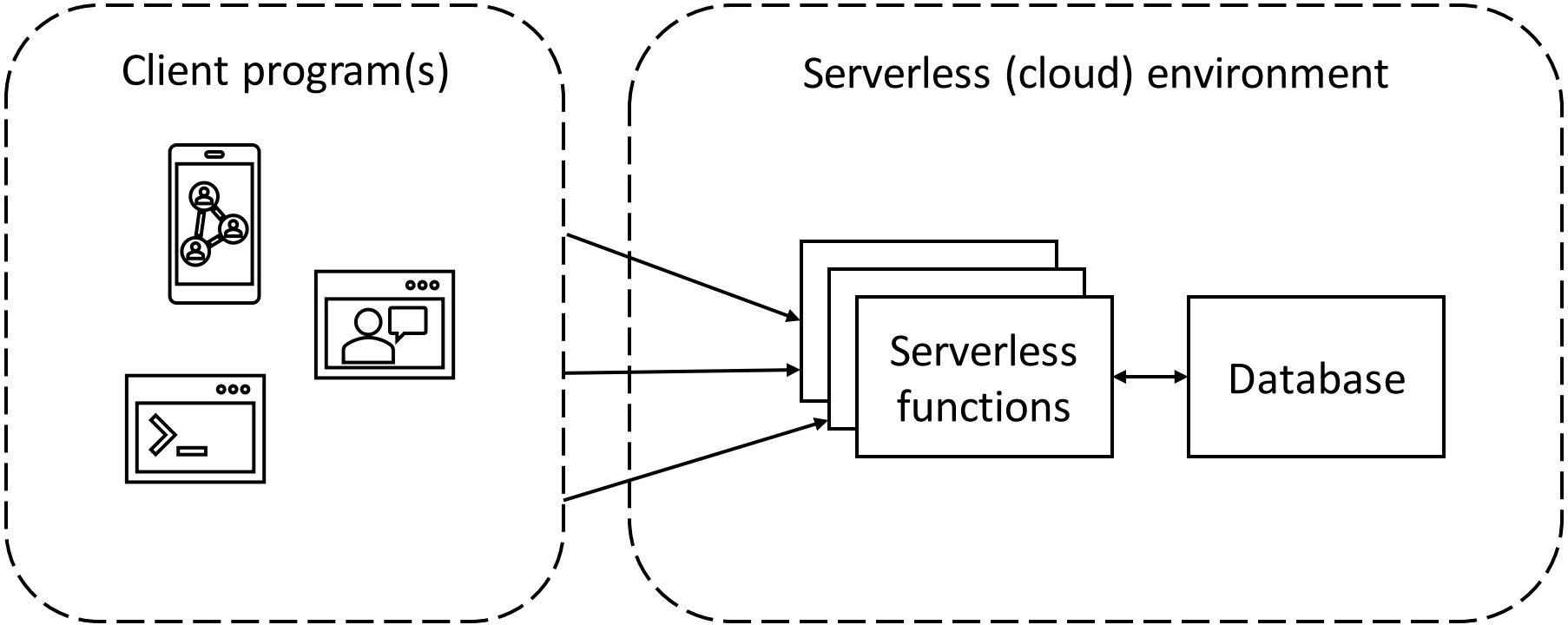
Advantages of serverless

- no need for servers' management
- automated scaling of services
- pricing based on actual resource usage

Function as a Service



Function as a Service



The problem

The developers need to write the code for interactions between the application logic and storage.

As a result, the approach:

- increases complexity
- slows down the development process
- hampers modularity and reuse

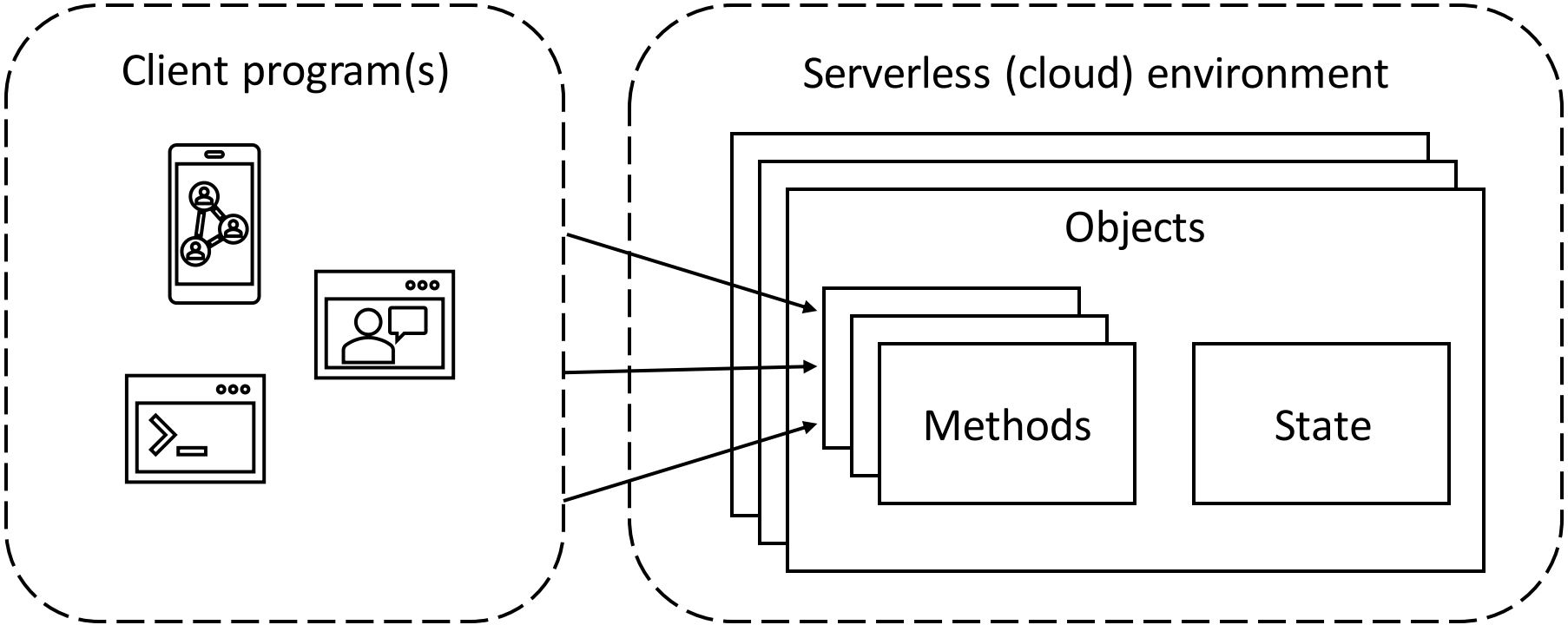
Nubes and its goals

Nubes is a novel programming model for stateful serverless functions.

The goal is to:

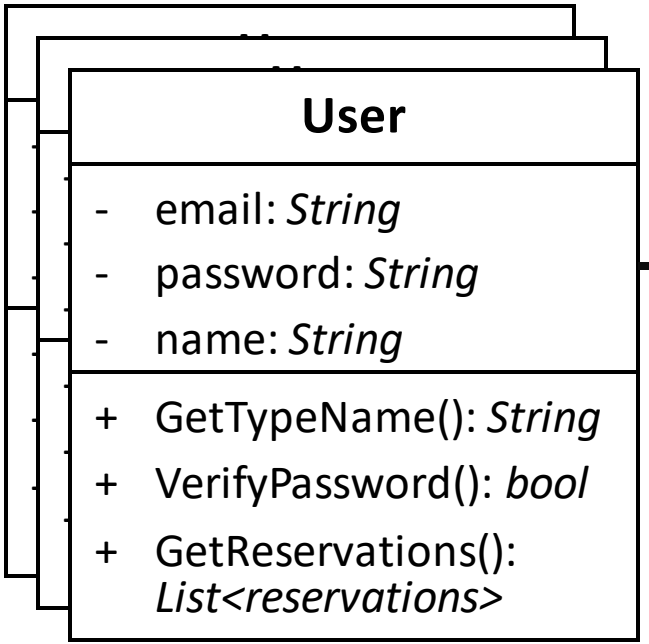
- accelerate the development process
- introduce low overhead in terms of latency

Objects as basic building blocks



Nubes code translation & generation

Object types

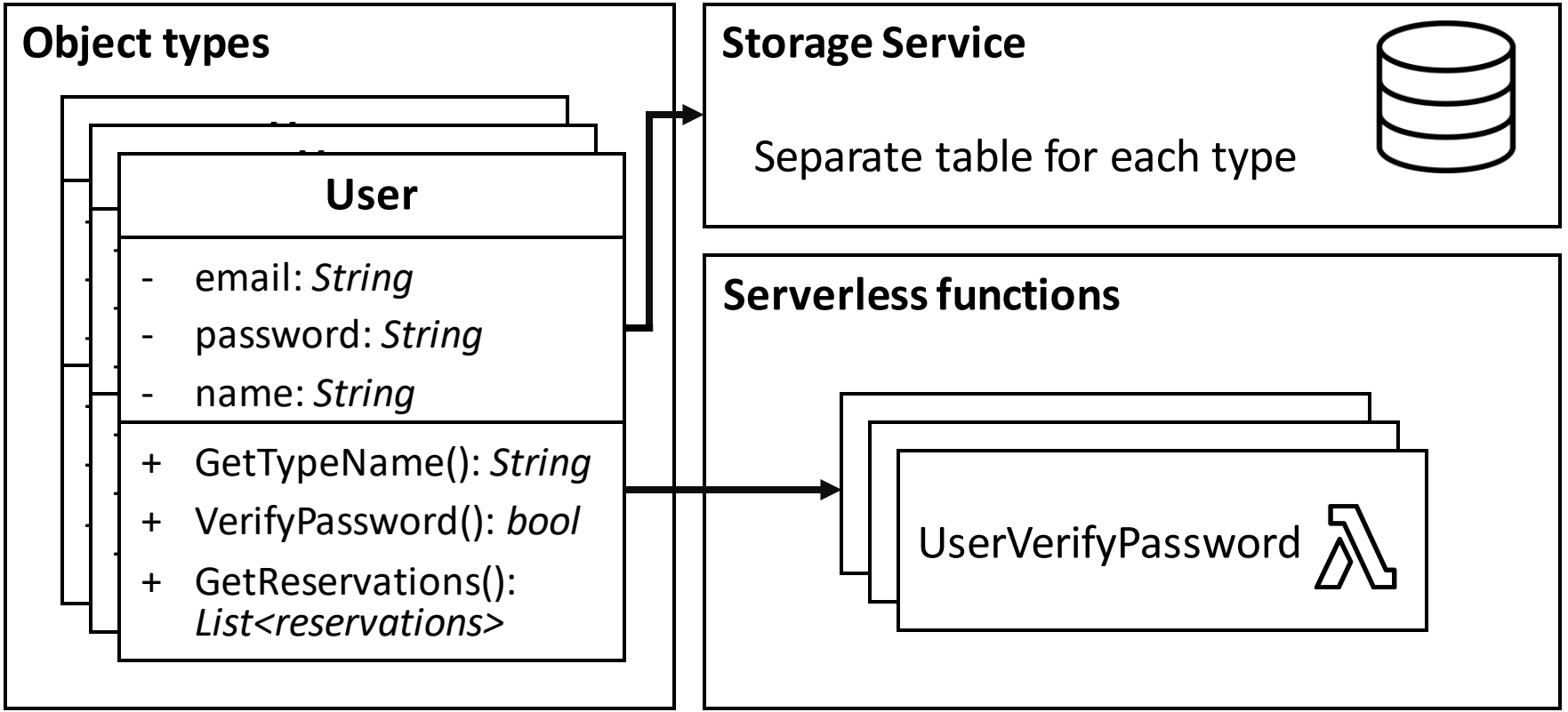


Storage Service

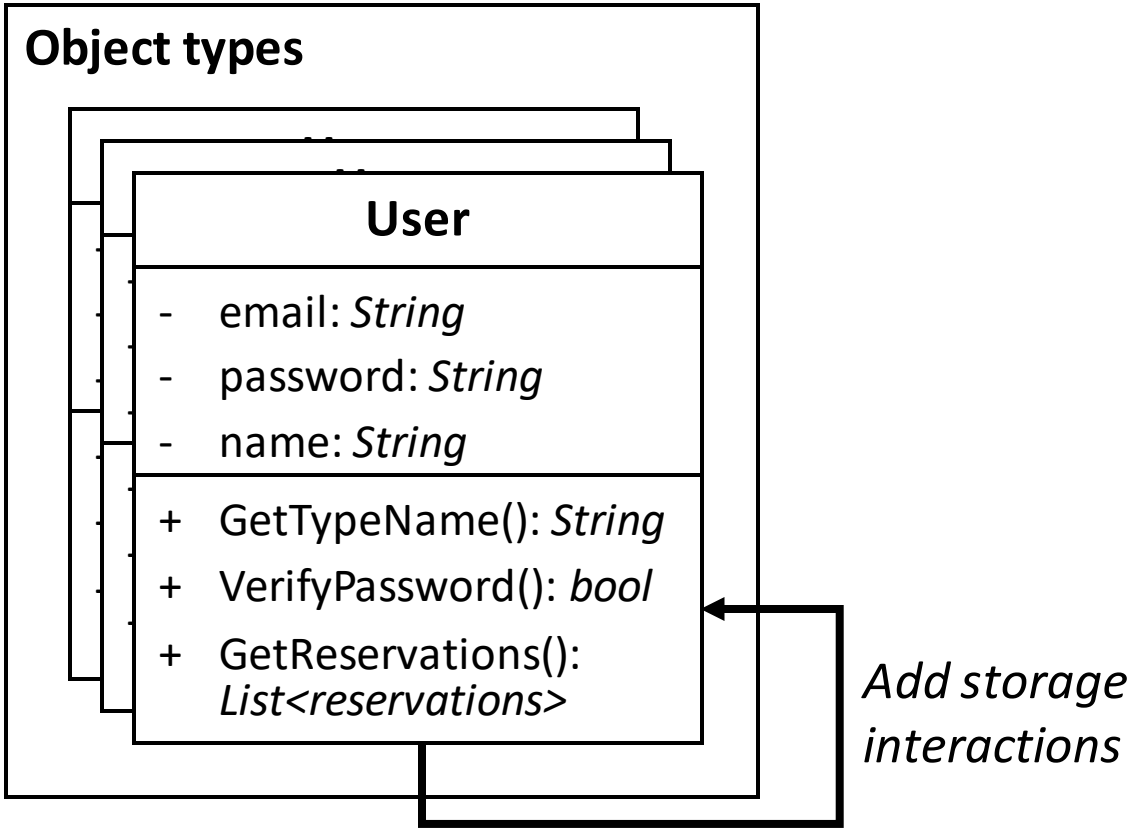
Separate table for each type

Email	Password	Name
kinga@gmail.com	dxbOLdPac4jV1	Kinga
marco@mail.com	aMlYDmnejvZL2	Marco
luca@polimi.it	vPRJq8xLJ7YJ6	Luca

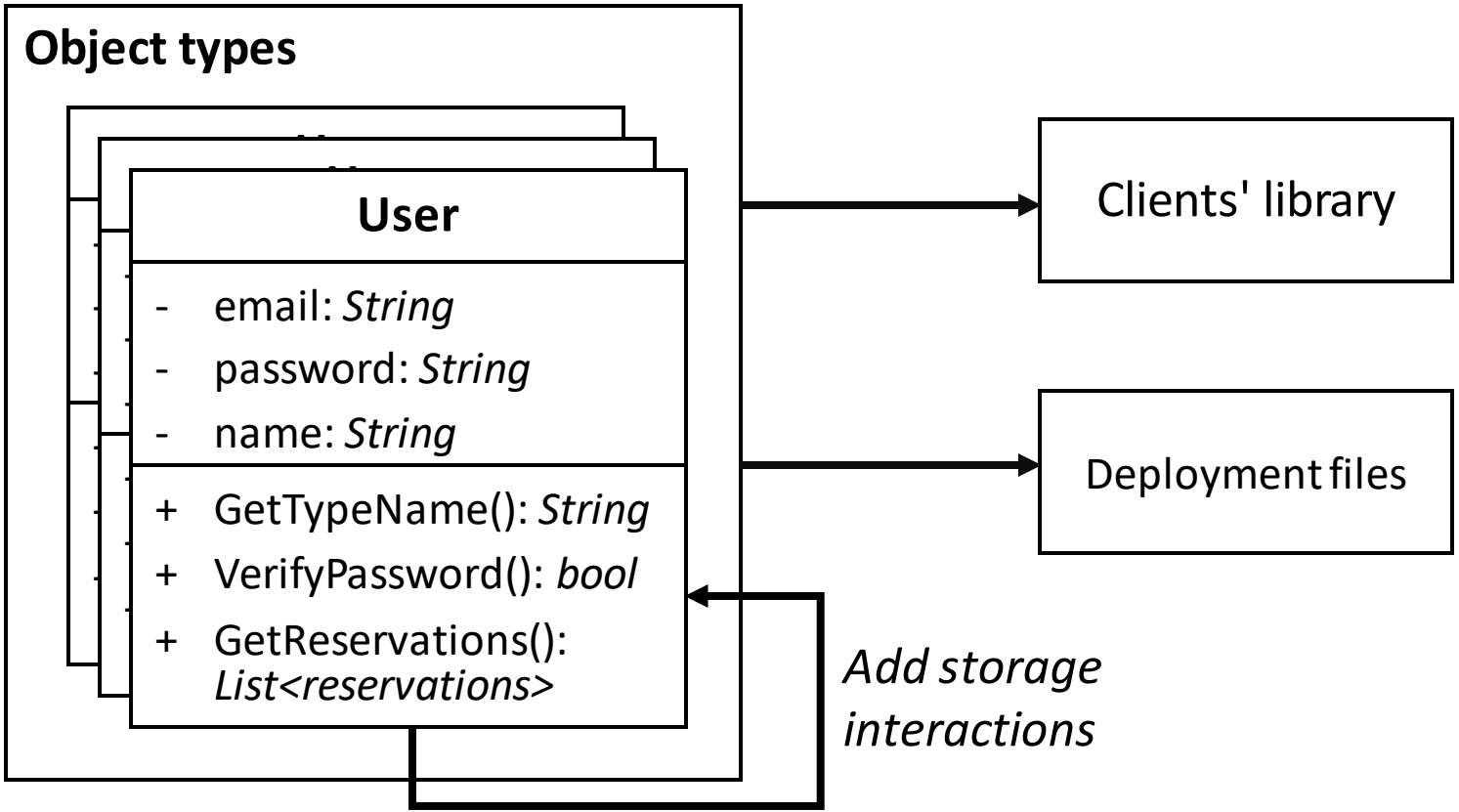
Nubes code translation & generation



Nubes code translation & generation



Nubes code translation & generation



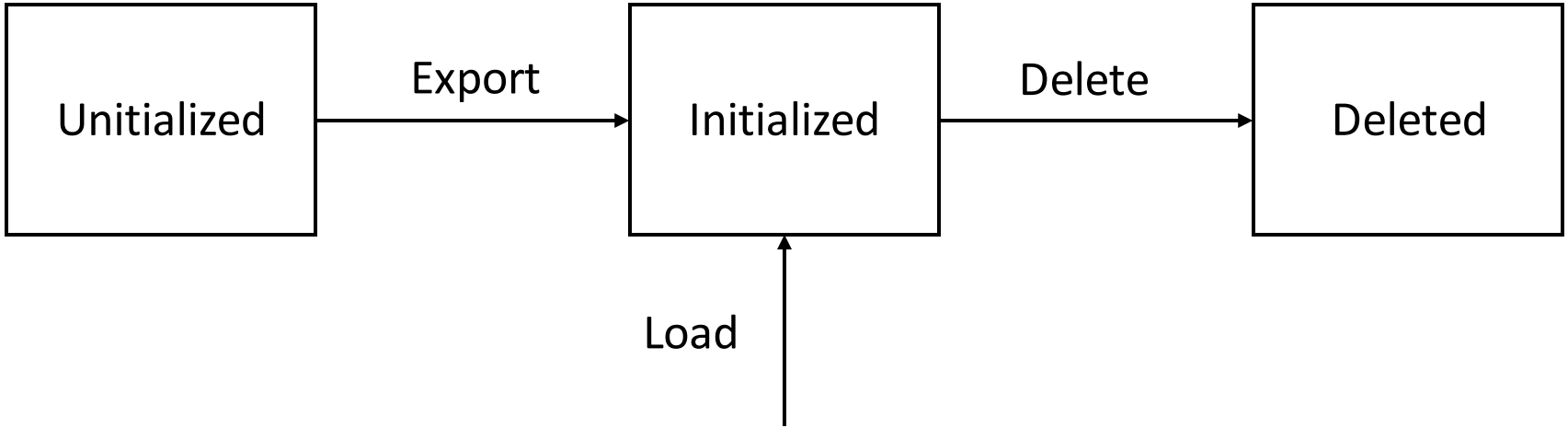
Object types' definitions

```
type User struct {
    Id                  string
    FirstName           string
    LastName            string
    Email               string
    Password            string
}

func (User) GetTypeName() string {
    return "User"
}
```

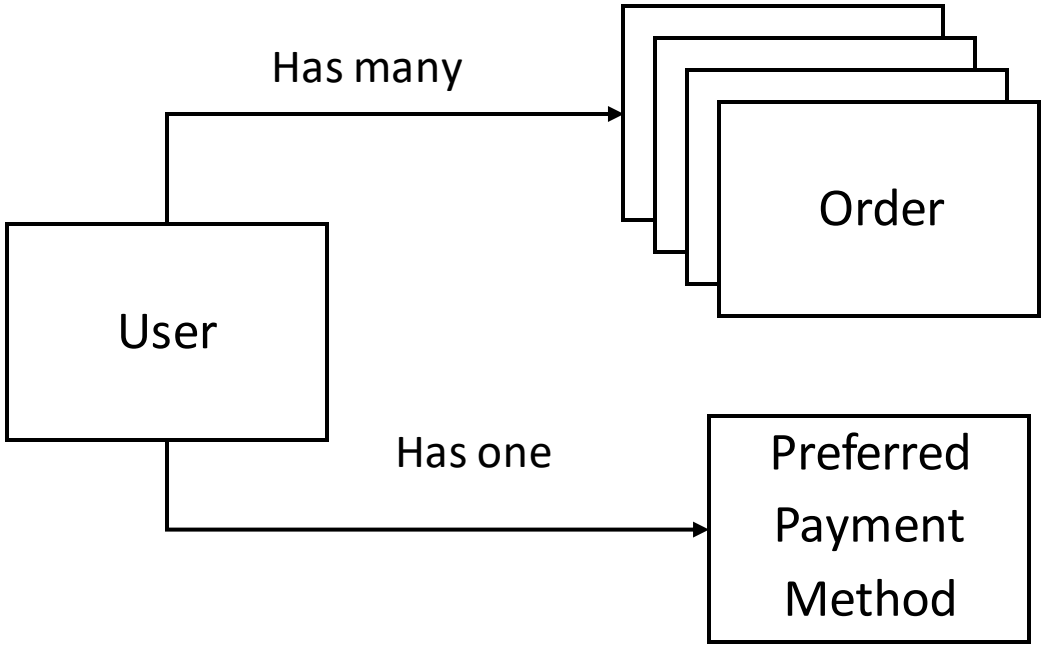
Custom Ids

```
type User struct {
    FirstName           string
    LastName            string
    Email               string `nubes:"id"`
    Password            string
}

func (User) GetTypeName() string {
    return "User"
}
```

Objects lifecycle



Relationships



Relationships

```
type User struct {
    FirstName           string
    LastName            string
    Email               string `nubes:"id"`
    Password            string
    Shops               lib.BiRefList[Shop]
     `nubes:"hasMany-Owners"`
    Orders              lib.RefList[Order]
}
```

Evaluation

Nubes was compared with standard methodology for developing stateful serverless applications: manual writing of interactions with the storage

Evaluation aimed at answering two research questions:

1. Are applications developed with Nubes simpler than equivalent baseline applications?
2. Does developing applications with Nubes incur any significant runtime performance overhead with respect to applicable baseline?

Experimental setup

Three implementations of server-side application for hotel booking service:

1. **Nubes**
   the proposed approach
2. **SSF**
   the traditional approach, normalized storage
3. **SSF-custom**
   the traditional approach, storage schema optimized for task at hand

Use cases were derived from *DeathStarBench* (Yu Gan et al., 2019), an open-source benchmark suite for microservices applications.

Effectiveness of the programming model

Concerns that were removed with Nubes:

- storage initialization
- read and write interactions with storage
- serverless functions handlers
- deployment files

System	Lines of code (written)	Lines of code (generated)	Lines of code (total)
SSF	1020	0	1020
Nubes	368	603	972

The baseline system requires **2.75 times** more lines of code than Nubes.

Discussion

Are applications developed with Nubes simpler than equivalent baseline applications?

Nubes:

- significantly reduces the amount of code to be written as well as its complexity
- enables a rapid migration to different cloud environments in the future

Experimental setup

- 50k users
- 5 cities
- 500 hotels
- 12.5k rooms
- 250k reservations

Task	Features tested
register-user	export
delete-user	delete
set-hotel-rate	update
login	object method
get-hotels	get all 1:n rel.
recommend	get inverse 1:n rel.
reserve	export, update, n:m rel.
get-user-reservations	get all 1:n rel.

Performance metric: the execution time of each serverless function.

As a metric of performance, the execution time of each serverless function involved in the application was measured.

Performance evaluation

- the results of the three workloads configurations are **nearly identical**
- on average, the overhead introduced by Nubes is within **10%** of the median duration of the SSF baseline
- the maximum overhead is about **23%** of the median w.r.t. SSF-custom system (task: retrieval of hotels in a city)
- the mean overhead considering all tasks w.r.t. SSF-custom is below **15%**

Discussion

Does developing applications with Nubes incur any significant runtime performance overhead with respect to applicable baseline?

- Nubes, is a general framework, thus it cannot provide custom, task-specific optimizations:
  - this kind of optimizations may bring advantages in some use cases, but they lead to less general and reusable components
- developers may adopt Nubes to speed up the development process and *then* manually optimize only the required parts (if any)
- Nubes preserves the typical scalability of serverless solutions

Conclusions

Nubes:

- promotes modularity and reusability through the usage of object-oriented programming concepts
- accelerates and simplifies the development process
- The overhead that is introduced is limited w.r.t. custom implementations that sacrifice generality and reusability for performance
- preserves scalability of serverless solutions

In the future, we plan to extend Nubes with mechanisms that synchronize concurrent access to the object's state.