

미디어프로젝트

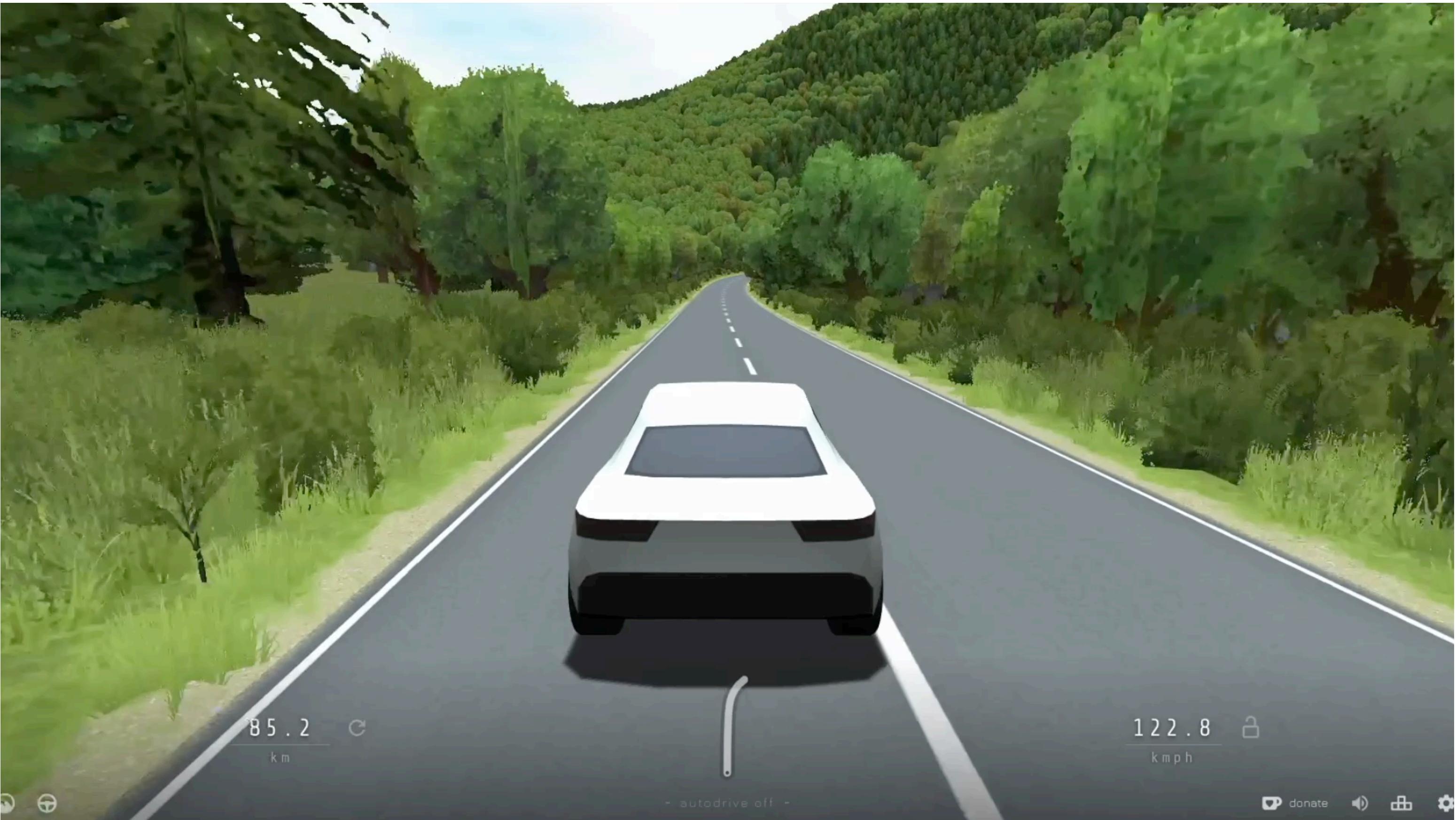
Zap: 멀티 디바이스 앱 라이브러리

팀 명 Zap

지도교수 오상은/소프트웨어학과

팀 원 박성범/디지털미디어학과





- 한 명의 사용자가 보유한 스마트 기기 수가 꾸준히 증가함에 따라 기기간 유기적 결합이 중요해지고 있음.
- 모바일 기기에는 동작 센서, 생체 센서, 터치스크린 등 유용한 장치가 다수 탑재되어 있음.
- 하지만, 장치들이 단일 기기에 종속되어 있어 활용이 어려움.

Zap: 멀티 디바이스 앱을 위한 애플리케이션 프로그래밍 라이브러리
하나의 기기에 종속된 한계를 극복하기 위한 프로그래밍 라이브러리 'Zap'을 제안.

원격 리소스
추상화

크로스 플랫폼
사용자 경험 실현

Zap: 멀티 디바이스 앱을 위한 애플리케이션 프로그래밍 라이브러리

하나의 기기에 종속된 한계를 극복하기 위한 프로그래밍 라이브러리 'Zap'을 제안.



원격 리소스
추상화

애플리케이션 계층에 추상화된 인터페이스를 제공함으로써 로컬 기기의 리소스에 접근하듯이 원격 기기의 리소스에 접근할 수 있어야 한다.

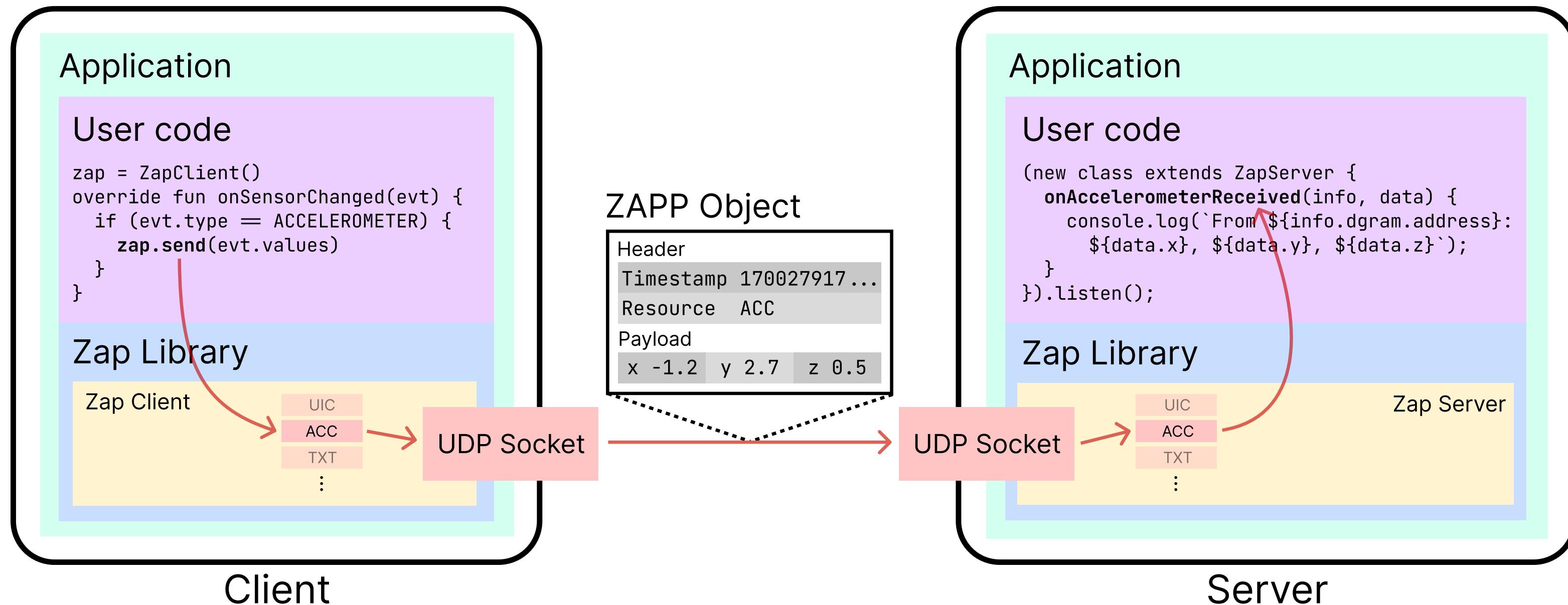
Zap: 멀티 디바이스 앱을 위한 애플리케이션 프로그래밍 라이브러리

하나의 기기에 종속된 한계를 극복하기 위한 프로그래밍 라이브러리 'Zap'을 제안.

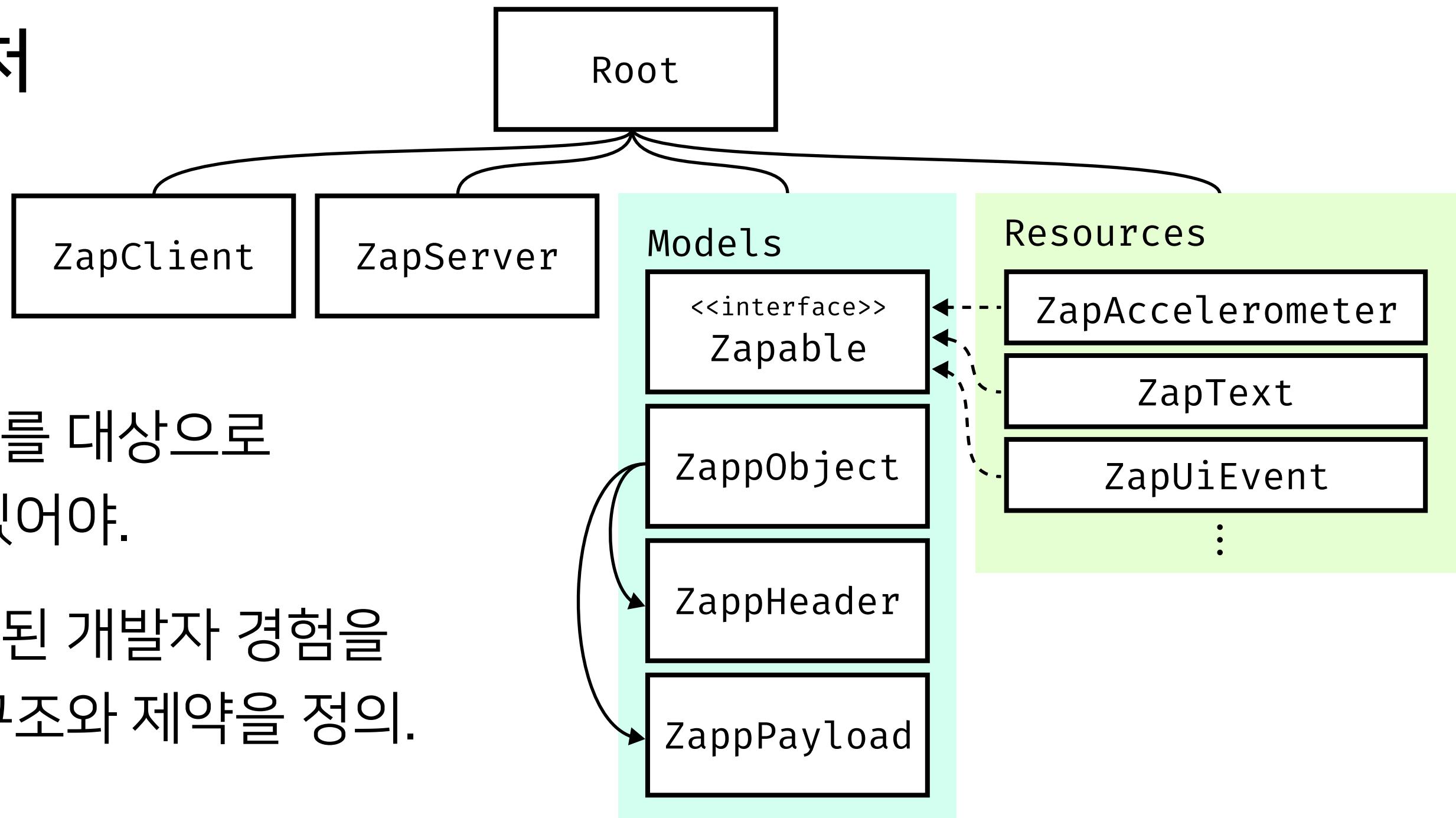
PC, TV, 키오스크와 같은 기기에는 모바일 기기에 탑재된 것과 같은 장치가 없으므로, 크로스 플랫폼을 지원해 확장된 멀티 디바이스 사용자 경험을 실현해야 한다.

크로스 플랫폼
사용자 경험 실현

- Client-Server 모델로 구조를 단순한 형태로 유지, 1:N 통신 지원.
- 기존 앱을 크게 변경하지 않고 멀티 디바이스 앱으로 전환할 수 있도록 구성.

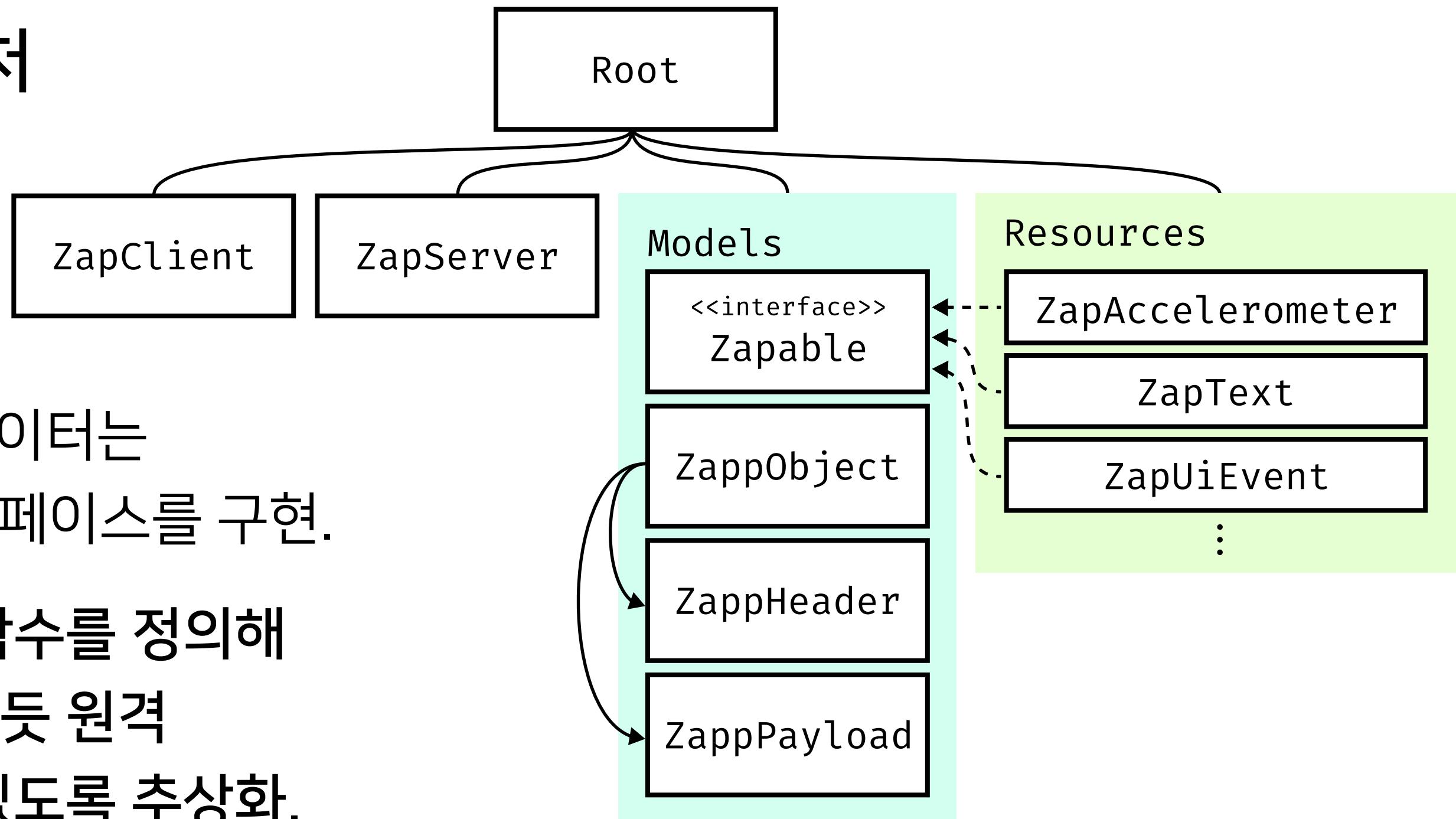


라이브러리 아키텍처



- 다양한 런타임과 언어를 대상으로 구현체를 개발할 수 있어야.
- 모든 구현체에서 일관된 개발자 경험을 제공하기 위해 표준 구조와 제약을 정의.

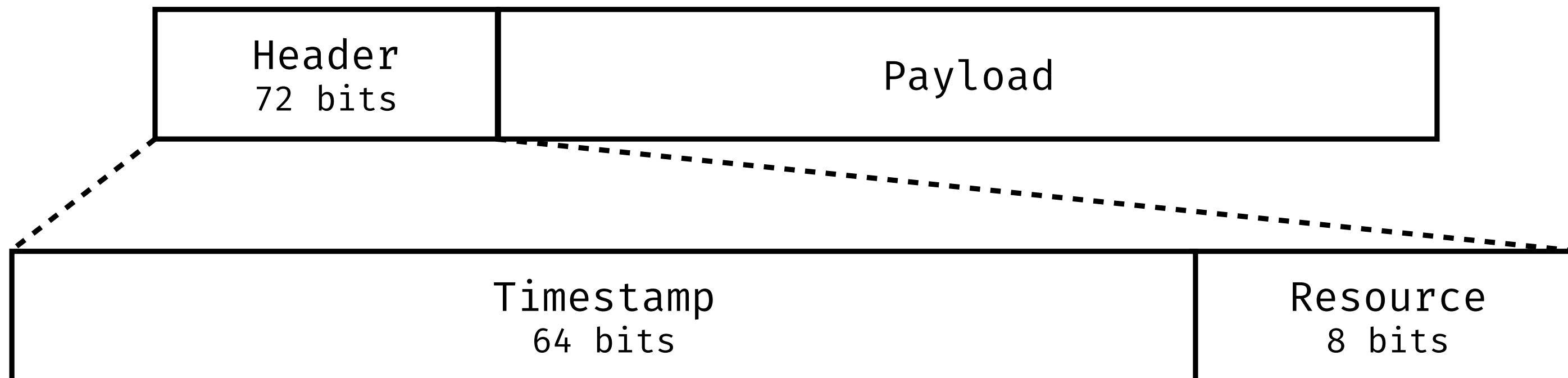
라이브러리 아키텍처



- Zap으로 전송되는 데이터는 반드시 Zapable 인터페이스를 구현.
- 서버 측에서는 콜백 함수를 정의해 로컬 리소스에 접근하듯 원격 리소스에 접근할 수 있도록 추상화.

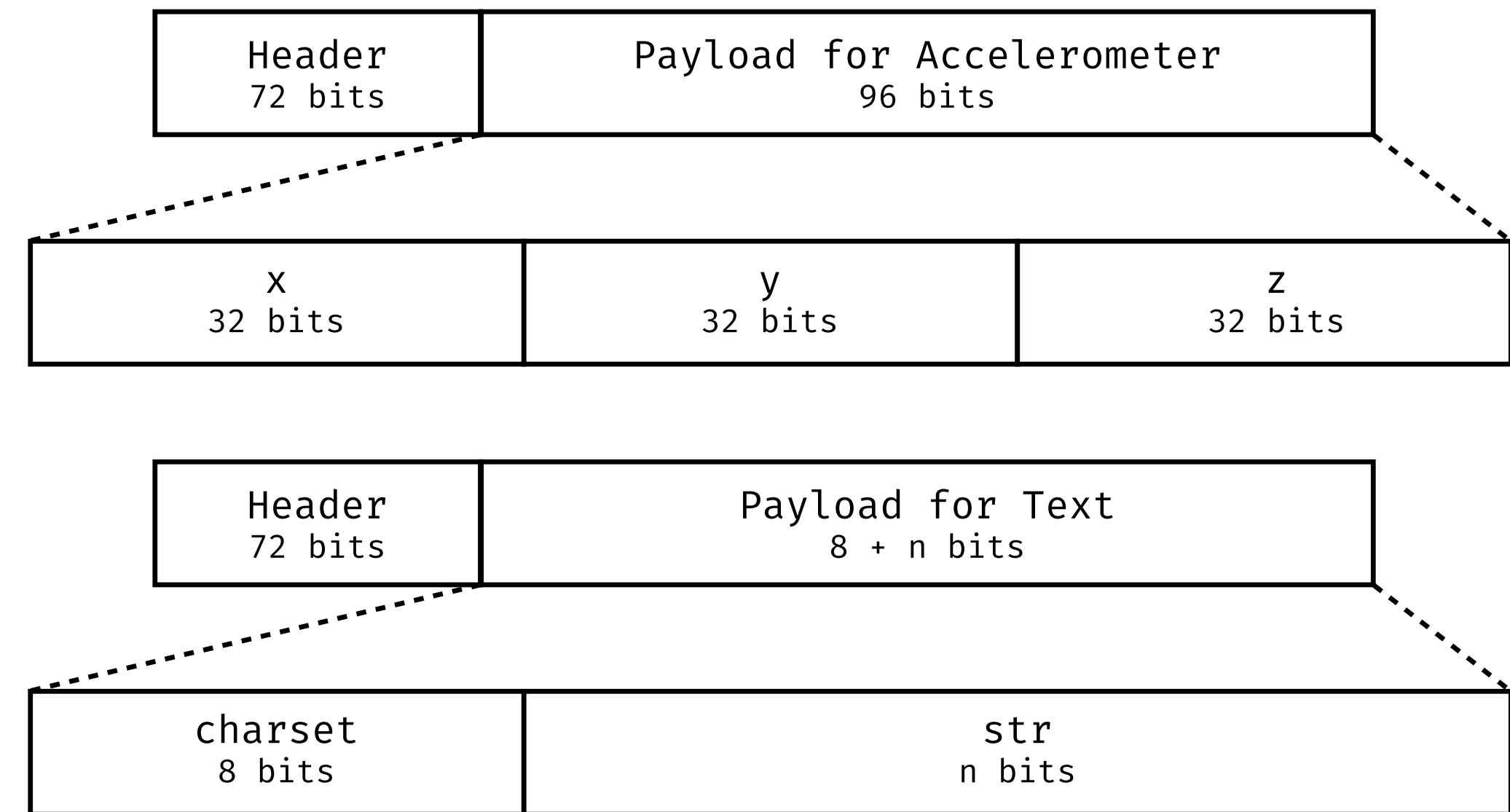
ZAPP (Zap Protocol)

- 높은 성능을 달성하기 위해 UDP 위에 자체적인 프로토콜 'ZAPP'를 정의.
- 9바이트 헤더 파트와 이어지는 페이로드 파트로 분리.
- 헤더 파트는 8바이트 타임스탬프 필드와 1바이트 리소스 필드로 구성.



ZAPP (Zap Protocol)

- 페이로드 파트의 형식은 리소스에 따라 상이.
- 헤더의 1바이트 리소스 필드를 참조해 페이로드의 형식을 파악할 수 있음.
- 플랫폼 독립적인 ZAPP 규격에 따라 원격 기기와 데이터를 주고 받을 수 있음.



Kotlin 및 Node.js 구현체 개발

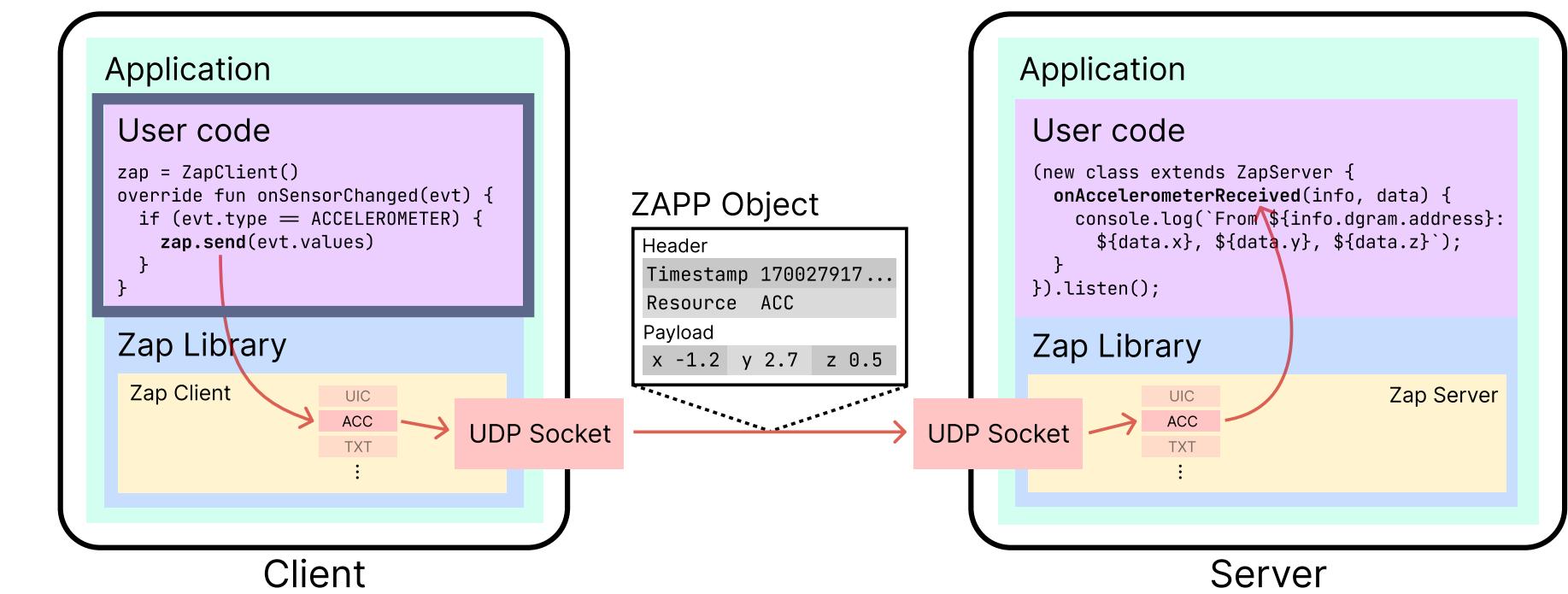
```

class MainActivity: AppCompatActivity(), SensorEventListener {
    private lateinit var zap: ZapClient

    override fun onCreate(state: Bundle?) {
        zap = ZapClient(InetAddress.getByName(...))
    }

    override fun onSensorChanged(event: SensorEvent) {
        if (event.sensor.type == Sensor.TYPE_ACCELEROMETER) {
            val (x, y, z) = event.values
            zap.send(ZapAccelerometer(x, y, z))
        }
    }
}

```

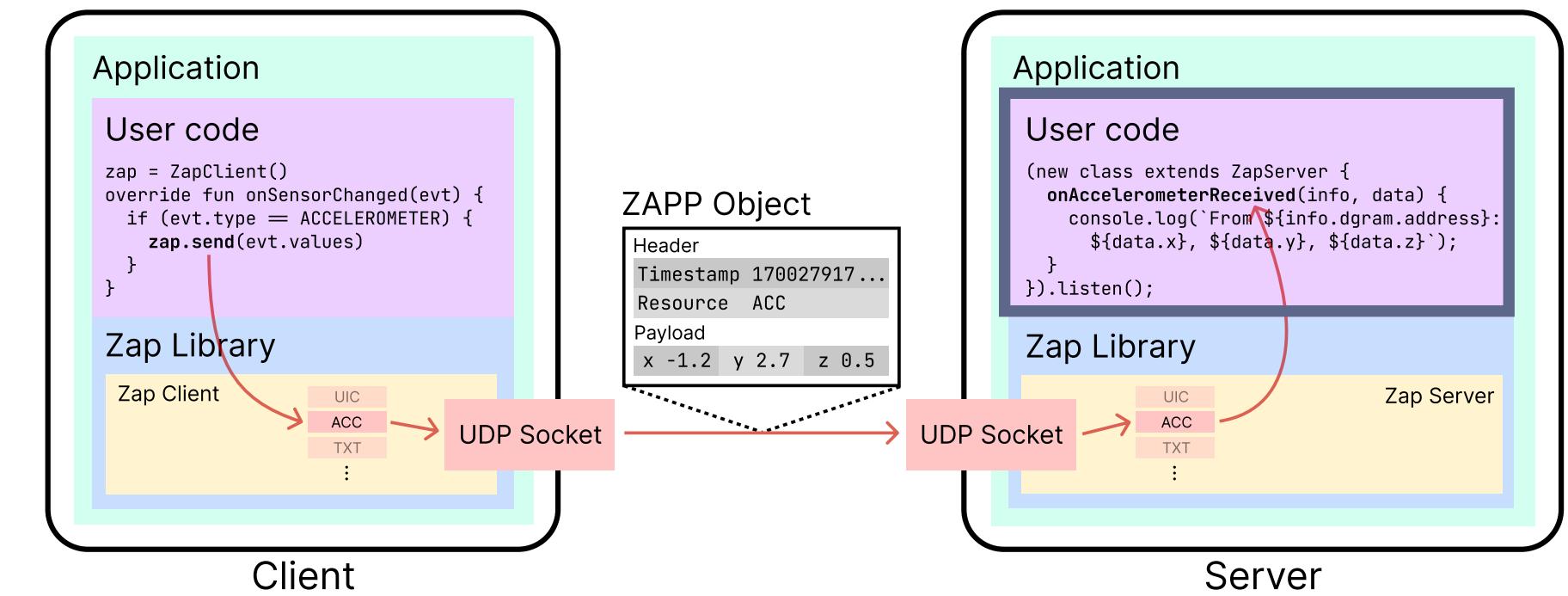


Kotlin 및 Node.js 구현체 개발

```
import { ZapServer } from 'zap-lib-js';
```



```
(new class extends ZapServer {
  onAccelerometerReceived(info, data) {
    console.log(`Data received:
      ${data.x}, ${data.y}, ${data.z}`);
  }
}).listen();
```



지원 리소스

- 가속도 센서 (ZapAccelerometer)
- 중력 센서 (ZapGravity)
- 자이로스코프 센서 (ZapGyroscope)
- 조도 센서 (ZapIlluminance)
- 자기장 센서 (ZapMagneticField)
- 지리적 위치 (ZapGeoPoint)
- UI 이벤트 (ZapUiEvent)
- 텍스트 (ZapText)

문서화 (zap-lib.github.io)

Zap

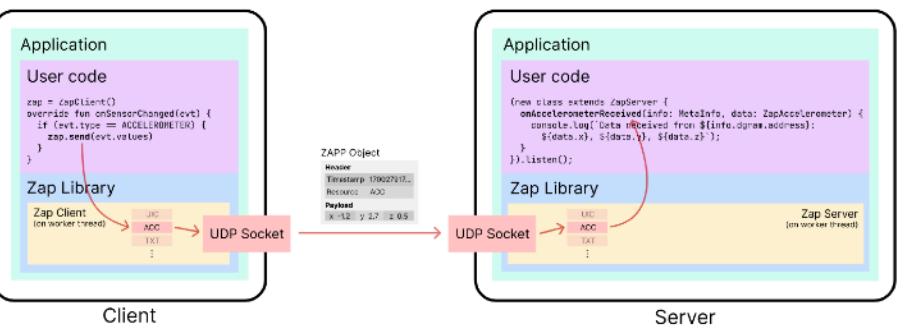
- Introduction
- Getting Started
- Architectures
- Specifications
- Implementations
- GitHub ↗

Search

Architectures

Architectures
Project Structures
ZAPP (Zap Protocol)

Zap is structured as a client-server model. The client unidirectionally sends data, and the server receives and processes it. This architecture keeps Zap's design simple and naturally allows a 1:N structure where multiple clients can send data to a single server.



A data sent from the client to the server is transmitted over UDP socket. Zap is implemented to send this data by defining 'ZAPP Object' defined on top of datagram, which contain the timestamp, resource type, and the actual data. ZAPP Object is the data unit of ZAPP(Zap Protocol), for more detailed information about the protocol, please check the [ZAPP page](#).

Zap uses callback functions to enable the transformation of a single-device application into a multi-device application while maintaining its own structure. The client, after obtaining the data they wish to transmit through the data source access API provided by their development framework, simply needs to call `zap.send(...)`. The server, running a Zap server instance, only needs to define callback functions to specify what code to execute each time it receives data from the client.

← Previous
Getting Started

Next
Project Structures →

Zap

- Introduction
- Getting Started
- Architectures
- Specifications
- Implementations
- GitHub ↗

Search

Client and Server

ZapClient

A client sends data to server.

type	signature	description
property	server_address	An IP address of the device running ZapServer.
function	send(obj: Zapable): void	Send given Zapable object to the server.
	obj : An object to send.	
function	stop(): void	Close the socket.

Table of contents
ZapClient
ZapServer
MetaInfo

ZapServer

A server receives data from client. The 'open' function refers to a callback function that users SHOULD override when declaring a ZapServer object to define its behavior. Refer to the Resources section for information on the parameters of each open function.

type	signature	description
function	listen(port: int): void	Start listening the transmitted data from clients on the given port.
	port : A port number for receiving data (default: 65500).	
function	stop(): void	Stop listening to clients.



패키징

- Node.js 구현체는 npm 레지스트리로, Kotlin 구현체는 JitPack으로 배포.
- Apache License 2.0 라이센싱 및 오픈소스 공개. (github.com/zap-lib)
- GitHub Action 워크플로우를 작성해 CI 구축.



Presentation Title

Presentation Subtitle

Author and Date

Try writing by hand on the client device.

- 플랫폼 독립적인 원격 리소스 공유 API와 프로토콜을 제공.
- 모바일-모바일 결합이 아닌 크로스 플랫폼을 지원하는 공개된 솔루션을 제시함으로써 기존 멀티 디바이스 연구와 차별성을 확보.
- 단일 기기의 한계를 극복해 멀티 디바이스 프로그램의 무궁무진한 가능성을 실현할 수 있을 것으로 기대.

- Naser AlDuaij and Jason Nieh. 2021. Tap: an app framework for dynamically composable mobile systems. In *Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys 2021)*. Virtual, 336–349.
- Naser AlDuaij, Alexander Van't Hof, and Jason Nieh. 2019. Heterogeneous Multi-Mobile Computing. In *Proceedings of the 17th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys 2019)*. Seoul, South Korea, 494–507.
- Sangeun Oh, Ahyeon Kim, Sunjae Lee, Kilho Lee, Dae R. Jeong, Steven Y. Ko, and Insik Shin. 2019. FLUID: Flexible User Interface Distribution for Ubiquitous Multi-Device Interaction. In *Proceedings of the 25th Annual International Conference on Mobile Computing and Networking (MobiCom 2019)*. Los Cabo, Mexico, 1–16.
- AirConsole, www.airconsole.com.
- Swip.js, github.com/paulsonnentag/swip.