# POSiOS – LiteServer en API

## Termen

### Wat is een WAR bestand?

Onze code wordt door middle van een WAR bestand op de LiteServer van de klant gezet. Een WAR bestand, ofwel Web application ARchive) is een JAR bestand dat gebruikt wordt om een verzameling JavaServer Pages, Java Servlets, Java klassen en XML bestanden te distribueren. Verder horen hier nog tags, bibliotheken, statische webpagina’s en andere benodigdheden om een webapplicatie te creëren.

### Wat is een JAR bestand?

JAR, ofwel Java ARchive, is een verpakkingsformaat welke typisch gebruikt wordt om Java klassen en geassocieerde metadata en benodigdheden te aggregeren tot één enkel bestand. Dit bestand kan vervolgens gebruikt worden om applicaties of bibliotheken op een Java platform te draaien.

### Wat is een API?

Een API, ofwel Application Programming Interface, is een verzameling definities op basis waarvan een computerprogramma kan communiceren met een ander programma of onderdeel (meestal in de vorm van bibliotheken). Vaak vormen API’s de scheiding tussen verschillende lagen van abstractie, zodat applicaties op een hoog niveau van abstractie kunnen werken en het minder abstracte werk uitbesteden aan andere programma’s. Hierdoor hoeft bijvoorbeeld een tekenprogramma niet te weten hoe het de printer moet aansturen, maar roept het daarvoor een gespecialiseerd stuk software aan in een bibliotheek, via een afdruk-API.

## Wat doet de LiteServer?

De LiteServer is een kleine server die de WAR lokaal draait. Normaal gezien maakt de Restaurant Butler applicatie op de iPad connectie met een van onze servers verspreid wereldwijd. Dit vormt op zich geen problemen, maar heeft echter twee minpunten:

1. Wanneer het internet wegvalt kan de applicatie niet meer werken.
2. De informatie staat op de servers en is dus niet eenvoudiger op te vragen door externe partijen zoals FOD.

Om deze redenen hebben we een LiteServer. Door middel van een geregistreerde blackbox verhelpt het bovenstaande problemen door:

1. De LiteServer fungeert als buffer tussen de servers en de klant.
2. Alle informatie passeert langs de blackbox en kan daarom eenvoudig geraadpleegd worden door externe partijen.

### Hoe werkt de LiteServer?

De LiteServer fungeert als buffer tussen de Restaurant Butler applicatie en onze servers. Doordat alle informative langs deze mini-server passeert kan alles eenvoudig tijdelijk opgeslagen worden. Wanneer het internet bijvoorbeeld wegvalt slaat de LiteServer de gegevens gewoon simpelweg lokaal op. Van zodra het internet terug is wordt deze informatie weer doorgespeeld naar de echte servers. Op deze manier is het onmogelijk om informatie te verliezen door bijvoorbeeld een slecht netwerk en kan de klant verder werken zonder hier problemen van te ondervinden.

De blackbox die gekoppeld is aan de LiteServer registreert alle handelingen. Doordat de blackbox aan de LiteServer gekoppeld is zit het rechtstreeks in verbinding met de bron van informatie. Alle informatie passeert langs de LiteServer, dus bijgevolg ook langs de blackbox.

## Hoe werkt de API?

De API wordt door ons verstrekt aan partners die met ons systeem willen integreren. Enkele voorbeelden zijn Dixys, Cashbox, LodgeGate, … Deze partijen dienen op voorhand een NDA (Non Disclosure Agreement) te tekenen alvorens zij onze API krijgen. Dit komt doordat ze inzage krijgen in de capaciteiten van ons systeem wat vertrouwelijke informatie is. Code zien ze echter nooit te zien.

Zie hier een voorbeeld van hoe een API werkt:

1. Stel een methode bestaat met als naam printNaam. Deze zou er als volgt uit zien:

*public void printNaam(String naam) {*

*System.out.println(naam);*

*}*

1. Deze methode krijgt als parameter een naam mee en print deze vervolgens af. Dit is natuurlijk een heel eenvoudige methode, maar stel dat deze methode enkele honderden lijnen code bevat. POSiOS creëert de methode en geeft vervolgens de methode in een API. De API method zal er als volgt uitzien:

*printNaam(java.lang.String)*

*printNaam neemt de naam over die meeggeven wordt als parameter en print deze uit.*

1. De partner krijgt dus de naam van de methode en de opties die hij dient mee te geven aan de methode plus een beschrijving van wat de functie doet.

U begint waarschijnlijk al een beeld te krijgen over hoe het in zijn werk gaat. Laten we het nu iets geavanceerder maken en kijken naar een actuele methode, bijvoorbeeld: addTable. De methode ziet er als volgt uit in onze code:

*public String addTable(String apiToken, int companyId, Table table) {*

*ApiUtil.checkIfCompanyIsReferredByUser(apiToken, companyId);*

*return tf.addTable(companyId, table);*

*}*

De API documentatie ziet er als volgt uit:

*addTable(java.lang.String apiToken, int companyId, Table table)*

*Adds a table to the given company.*

De partner weet dus dat hij een token moet meegeven (een token is een unieke tijdelijke sleutel die gegenereerd wordt langs onze kant om de partner te identificeren), een nummer dat het bedrijf identificeert en een tafel. Verder heeft de partner geen inzicht in wat de methode doet behalve wat wij meegeven als beschrijving, in dit geval: “Adds a table to the given company.”

### Waarom is een API effectief?

Een partner roept een methode aan die zich op de bovenste laag bevindt van de software. De software (dat zicht bevindt op de server) heeft meerdere lagen gaande van de interface laag tot de laagste laag namelijk de laag waar connecties gemaakt worden met de database.

Doordat de partner enkel methodes kan aanroepen op het bovenste niveau hebben ze verder geen inbreng op wat er gaat gebeuren. Zoals u misschien reeds gezien heeft gaat de methode addTable eerst een controle uitvoeren. De functie ApiUtil.check… controleert eerst of het bedrijf in kwestie wel degelijk gekoppeld is aan de account van de partner. Het spreekt voor zicht dat een partner zomaar niet een ander zijn of haar account mag wijzigen. In lagere lagen controleren we dan of de informatie in orde is die we krijgen aangezien we er nooit van kunnen uitgaan dat de informatie die we krijgen in orde is.

### Authenticatie

Hoe zit het nu met die unieke sleutel en authenticatie? Bij partner integratie, na het tekenen van een NDA, krijgt de partner een gebruikersnaam en wachtwoord ter beschikking op onze development server. Vervolgens koppelen we een test account aan hun account zodat ze effectief kunnen testen hoe het in een productie omgeving er aan toe zou gaan: partner en klant.

Door middel van de gebruikersnaam en wachtwoord kan de partner een apiToken krijgen. Een apiToken is een unieke sleutel dat geldig is voor vier uur. Deze token kan gebruikt worden om methodes aan te roepen via onze API. De eerste controle die gebeurt van onze kant is kijken of de sleutel al dan niet geldig is. Als die geldig is gebeurt de volgende controle namelijk kijken of de partner wel degelijk een wijziging mag aanbrengen aan de account die zij gespecifieerd hebben.

Na verloop van tijd verloopt deze sleutel en dienen ze een nieuwe sleutel aan te vragen. Van zodra de integratie voltooid is komt de testfase. Aangezien de ontwikkeling op een development server gebeurt (server met testdata) kan het geen kwaad dat er iets misloopt. Tijdens de testface gebeurt een interne audit door POSiOS zelf.

Tijdens de testfase gaan wij de integratie testen door hetzelfde stramien te volgen. Bijvoorbeeld als wij een integratie doen met een partner dat voor reserveringen zorgt, gaan wij reservaties aanmaken/wijzigen/… en kijken wat er allemaal gebeurt op onze server. Op deze manier zien wij hoe zij onze API hebben gebruikt en of er al dan niet overbodige aanroepingen zijn gebeurd van de methodes. Pas als dit in orde is laten wij hen live gaan op een live server van ons. Ook hiervoor heeft de partner een nieuwe gebruikersnaam en wachtwoord nodig.

De authenticatie is tweeledig aangezien de partner met een gebruikersnaam en wachtwoord nog niets kan doen. Wij dienen vervolgens een klant aan hun account te koppelen. Bijvoorbeeld: een klant van POSiOS is klant van het bedrijf dat voor reserveringen zorgt. Doordat zij wijn geïntegreerd met ons kunnen wij simpelweg de klant koppelen aan de partner op de live server. Pas dan is de partner geauthentiseerd om wijzigingen door te voeren aan de klant hun account.

### Nut van een API

U ziet natuurlijk het nut van een API. Wij hebben de mogelijkheid onze software uit te breiden door te integreren met partners zonder dat wij hen onze code dienen te geven. Wij kunnen ons dan specifiëren op onze niche terwijl de partner zich focust op zijn of haar niche.

Een API is ook een heel beperkt document waarin alleen de benodigde functies zitten die de partner nodig heeft. Het bedrijf dat voor reserveringen zorgt krijgt enkel de methodes om een reservatie toe te voegen en te wijzigen (verwijderen laten we niet toe). De API wordt zodanig beperkt gemaakt omdat we de partner niet meer middelen willen geven dan dat ze nodig hebben. Alle belangrijke functies blijven intern en stellen we niet beschikbaar.

Wij hopen u zodoende genoeg informatie gegeven te hebben om het nodige inzicht te verwerven in onze API en systeem.