Implementeren van Pydev in IBM Developer for z/OS om Python applicaties uit te voeren op de mainframe.

Alif Monnoye.

Scriptie voorgedragen tot het bekomen van de graad van
Professionele bachelor in de toegepaste informatica

Promotor: Dhr. L. Blondeel
Co-promotor: Dhr. N. Sletten
Instelling: Den Norske Bank (DNB)
Academiejaar: 2023–2024
Eerste examenperiode

Departement IT en Digitale Innovatie .

HO
GENT

Woord vooraf

Dit is een proof of concept die ik wil uitwerken omdat dit hulp kan bieden in de modernisatie van de mainframe systemen van IBM. Op het moment van schrijven heb ik al stage gedaan bij 2 bedrijven die gebruik maken van een mainframe en deze gebruiken allebei IBM Developer for zOS om applicaties te schrijven in verschillende soorten programmeertalen zoals COBOL, PL/1, Python, Java, ... Python is één van de meest gebruikte programmeertalen op de dag van vandaag en hierin wordt dus veel ingezet door IBM om deze programmeertaal zo efficiënt mogelijk te laten werken op hun systemen. De skills in de 'oudere' programmeertalen zoals COBOL en PL/1 is sterk aan het afnemen dus is de inbreng van Python bijna een must om nog goede programmeurs te vinden. Een goede editor is ook nodig wat momenteel niet het geval is in IDz dus maak ik in dit onderzoek een stappenplan om dit zo goed mogelijk op te stellen.

Ik zou Njaal Sletten en Stian Botnevik van DNB, Noorwegen willen bedanken voor de hulp en inspiratie bij het schrijven en uitwerken van dit onderzoek. Ze hielpen allebei met het technische gedeelte en Stian kwam met de probleemstelling.

Samenvatting

IBM heeft voor zijn IBM Z Systems de deur opengezet naar vernieuwing door Java en Python toegankelijk te maken op de Mainframe. Hoewel deze talen ondersteund worden, is het niet altijd even makkelijk om programma's hierin te schrijven. IBM Developer for z/OS is een stap in de goede richting door zijn functionaliteit om in een bekend programma scripts te openen en aan te passen die opgeslagen zijn op de mainframe. Aangezien dit Eclipse based is, is er geen Python interpreter waardoor Python scripts aangepast moeten worden in een text editor.

Deze paper zal een proof of concept zijn voor het opstellen van Pydev en een Python IDE in IBM Developer for z/OS. Hierdoor kunnen Python developers efficiënter code schrijven, aanpassen en submitten in de Unix System Services omgeving van de mainframe. Dit is belangrijk voor de programmeurs van DNB die werken in dit programma die nu hun Python code moeten testen in een ssh connectie.

In dit onderzoek zal er gewerkt worden met de Pydev plug-in voor Eclipse en zal er een stap voor stap instructie gegeven worden over hoe dit ingesteld moet worden. Ook zal er onderzocht worden of deze scripts uitgevoerd kunnen worden door IBM Developer for z/OS in de Mainframe omgeving.

Het resultaat zal een volledige Proof of concept zijn voor een Python interpreter op te zetten in dit programma. Dit is vooral relevant voor Python developers in bedrijven die IBM Developer for z/OS gebruiken in combinatie met een IBM Z Mainframe.

Inhoudsopgave

Lijst van figuren

1

Inleiding

1.1. Probleemstelling

IBM heeft voor zijn IBM Z Systems de deur opengezet naar vernieuwing door Java en Python toegankelijk te maken op de Mainframe. Hoewel deze talen ondersteund worden, is het niet altijd even makkelijk om programma's hierin te schrijven. IBM Developer for z/OS is een stap in de goede richting door zijn functionaliteit om in een bekend programma scripts te openen en aan te passen die opgeslagen zijn op een mainframe. Aangezien dit Eclipse based is, is er geen Python interpreter die gebruikt kan worden waardoor Python scripts enkel geopend en aangepast kunnen worden in een text editor. Dit komt vanzelfsprekend niet met een syntaxcheck, code completion, etc

Dit is een probleem voor developers zoals Stian Botnevik van DNB om Python code te schrijven in dit programma. Hoewel scripts vaak ook lokaal worden geschreven in een programma zoals Visual Studio Code en dan worden overgezet naar de mainframe, moeten er nogsteeds aanpassingen gebeuren door de verschillen in runtime environment. In een klein Python programma lukt dit wel nog in een text editor maar als ze wat groter zijn, wordt dit veel moeilijker.

1.2. Onderzoeksvraag

In dit onderzoek zal er gewerkt worden met de Pydev plug-in voor Eclipse en zullen er stap voor stap instructies gegeven worden over hoe dit opgezet wordt. Aangezien IBM Developer for z/OS toch nog verschillend is van Eclipse kunnen er verschillende problemen opkomen die niet gebeuren in het standaard Eclipse programma.

Eens dit is opgezet, zal er onderzocht worden of deze scripts uitgevoerd kunnen

worden in de Mainframe omgeving via IBM Developer for z/OS.

1.3. Onderzoeksdoelstelling

Het resultaat zal een volledige proof of concept zijn voor een Python interpreter op te zetten in IBM Developer for z/OS. Dit is vooral relevant voor Python developers in bedrijven die dit programma gebruiken in combinatie met een IBM Z Mainframe.

1.4. Opzet van deze bachelorproef

De rest van deze bachelorproef is als volgt opgebouwd:

In Hoofdstuk 2 wordt een overzicht gegeven van de stand van zaken binnen het onderzoeksdomein, op basis van een literatuurstudie.

In Hoofdstuk 3 wordt de methodologie toegelicht en worden de gebruikte onderzoekstechnieken besproken om een antwoord te kunnen formuleren op de onderzoeksvragen.

In Hoofdstuk 4 wordt de gebruikte terminologie in detail besproken samen met andere onderwerpen relevant voor het onderzoek.

In Hoofdstuk 5 wordt de proof of concept duidelijk weergegeven en besproken.

In Hoofdstuk 6 wordt de connectie met de USS omgeving opgezet in IDz.

In Hoofdstuk 7 wordt de runtime environment getest om Python scripts in uit te voeren.

In Hoofdstuk 8 worden de functies van Pydev getest door een Python API te schrijven.

In Hoofdstuk 9 zal de geschreven API getest worden in de IDz terminal.

In Hoofdstuk 10 zal de testmethode geëvalueerd worden.

In Hoofdstuk 11, tenslotte, wordt de conclusie gegeven en een antwoord geformuleerd op de onderzoeksvragen. Daarbij wordt ook een aanzet gegeven voor toekomstig onderzoek binnen dit domein.

2

Stand van zaken

2.1. Een andere manier van werken

Veel bedrijven die werken met een mainframe hebben het probleem dat ze onvoldoende mensen vinden met de nodige kennis over deze technologie. De introductie van Unix systemen op de mainframe in het jaar 2000 (Mertic, 2020) had dit probleem wat verminderd maar zeker niet weggewerkt. De oorzaak van deze situatie is door de oude technieken die het systeem gebruikt. COBOL en PL/1 zijn niet de meest aantrekkelijke programmeertalen om te leren en mensen kiezen liever Bash als scriptingtaal in plaats van REXX. Hoewel IBM veel inzet op documentatie en online leerplatformen zoals IBM Z Xplore, blijft het tekort van ervaren mensen nog steeds te laag.

De mainframe wereld zal zich dus moeten aanpassen aan de vaardigheden van de mensen door over te schakelen naar beter gekende manieren van werken. Python bijvoorbeeld is een programmeertaal die steeds meer populariteit krijgt sinds zijn ontstaan in de vroege jaren 90. Dit is niet enkel bij reeds ervaren programmeurs, maar mensen die net beginnen programmeren kiezen hier steeds vaker voor. Dit is vooral door zijn begin vriendelijkheid, verschillende doeleinden en een actieve community. (Johnson, 2023)

IBM heeft dit opgemerkt en een Python- compiler en interpreter ontwikkeld voor z/OS genaamd IBM Open Enterprise SDK for Python. Hierdoor kun je met Python interactie hebben met z/OS om bijvoorbeeld applicaties te ontwikkelen of de resources van het systeem beheren. Door de Unix omgeving heeft de programmeur ook geen geavanceerde z/OS kennis nodig. (Klaey, 2023)

2.2. Bedrijven in het werkveld

Een bank is een goed voorbeeld van een bedrijf die gebruik maakt van IBM hun Z Systems. Dit valt te concluderen door het feit dat maar liefst 92 van de top 100 banken wereldwijd een mainframe gebruikt. Dit is ook logisch aangezien er gemiddeld 12.6 miljard financiële transacties per dag zijn dat door deze systemen wordt verwerkt. (Wagle, 2017)

Door dit aantal is de nood voor een mainframe toch niet te onderdrukken maar het is niet enkel het aantal transacties dat deze machine kan uitvoeren, het is ook de snelheid, schaalbaarheid en beveiliging van deze systemen dat een grote rol speelt. Het zijn niet enkel banken die van deze technologie gebruik maken, maar ook verzekeringsmaatschappijen bijvoorbeeld. De top 10 van deze soort bedrijven maken allemaal gebruik van een IBM Mainframe (Tozzi, 2022)

2.3. De Skill gap

Het is nog steeds moeilijk om nieuw talent aan te trekken in de mainframe wereld. Een onderzoek van Deloitte (2020) toont aan dat 79% van de projectleiders moeilijkheden heeft met het zoeken naar mensen met de juiste skillset. Hetzelfde onderzoek toont ook aan dat er in de teams zelf een groot verschil is van kennis en vaardigheden.

Hoewel dit systeem gebruikt wordt door 71% van de fortune 500 companies (Tozzi, 2022) , is de Skill gap nog steeds te groot waardoor veel bedrijven vrezen voor een groot tekort aan werknemers om deze Z Systems te onderhouden.

Volgens Petra Goude (2023) zijn er verschillende manieren om dit probleem aan te pakken. Zo kunnen bedrijven lessen geven over Mainframe en hoe ze dit gebruiken. Ze vertelt ook dat de vaardigheden die nodig zijn beter gecomplimenteerd moeten worden en kunnen leiden naar een belonende en lange termijn carrière. Hoewel dit zou helpen, vind ze dit niet de kern van het probleem. Het zijn de oude technieken die mensen niet aantrekt. Ze stelt voor om meer te investeren in hedendaagse technologie en dit te installeren op de mainframe. Dit kan gaan over dezelfde test -en deployment technieken, maar ook over hedendaagse programmeertalen zoals Java of Python. Dit zou kunnen door middel van API's en zou een nieuwe, jongere werkkracht aantrekken.

2.4. Toch nog een kleurrijke toekomst

Ondanks deze probleemstelling, ziet de toekomst er toch nog goed uit voor deze systemen. Zo wordt er meer gemoderniseerd met bijvoorbeeld modernere programmeertalen. Er wordt ook voorspeld dat we een introductie van DevOps en

self-service benaderingen gaan zien. (Pennaz, 2023)

Momenteel zijn Python en Java beschikbaar als programmeertaal op de mainframe naast PL/1 en COBOL. Sinds deze introductie zijn al bijna 2/3de van gebruikers op de mainframe Java aan het toepassen op een bepaalde manier. (Watts, 2018)

3

Methodologie

3.1. Fase 1: Literatuurstudie

In deze fase zal er relevante informatie verzameld worden over relevante vakliteratuur. Dit zal gedaan worden door betrouwbare bronnen op te zoeken en te bestuderen.
De tijd die hiervoor gereserveerd wordt, is 5 weken en zal als resultaat een volledig overzicht geven van de omgeving waarin dit onderzoek zich bevindt.

3.2. Fase 2: Opstellen Pydev

Het doel in deze fase is het implementeren van Pydev in IDz door de opgezochte literatuur te bestuderen en te onderzoeken. Eens dit gedaan is, zal de implementatie van Pydev beginnen over een periode van 3 weken. Na deze implementatie zullen Python applicaties geopend kunnen worden met Pydev voor functies zoals code completion, syntax check en code analyse.

3.3. Fase 3: Runtime environment opzetten

In deze fase zal er een connectie gemaakt worden met de USS omgeving zodat de Python applicaties daar uitgevoerd kunnen worden via IDz. Dit zal 2 weken duren en gebruikers kunnen dan via IDz hun Python applicaties uitvoeren in de USS omgeving.

3.4. Fase 4: Testen van Pydev en connectie met USS

In de testfase zal er een testomgeving opgezet worden in USS via IDz om zo de mogelijkheden en limitaties van deze connectie te achterhalen. In IDz wordt er een Python API ontwikkeld om te testen hoe efficiënt Pydev is voor het programmeren in Python. Hiervoor zijn 2 weken vrijgehouden en zal als resultaat de voor -en

nadelen terug geven.

3.5. Fase 5: Evaluatie voorbereiding

In deze fase valt de conclusie van heel de bachelorproef en het voorbereiden op de eindevaluatie. Dit duurt 2 weken.

4

Literatuurstudie

4.1. De IBM Z Systems omgeving

Het is belangrijk om te weten wat een mainframe is en wat de hoofdpunten van de technologie zijn. Het is moeilijk om een goede definitie te plakken op deze term maar het is ontwikkelt door IBM en het wordt vooral gebruikt door grote bedrijven om belangrijke applicaties te hosten en/of veel transacties te kunnen doorvoeren. Hoewel dit ook mogelijk is op een kleinschalige server, zou het resultaat niet hetzelfde zijn omdat mainframes miljarden transacties per dag zou kunnen uitvoeren zonder enige vertraging. (BasuMallick, 2023)

De mainframe wordt vaak in vergelijking gebracht met een traditionele server die je vind in een datacenter. Deze vergelijking is niet onterecht omdat ze wel een gelijkaardige functie hebben. Een mainframe zoals de hedendaagse IBM Z16 heeft de mogelijkheid om 19 miljard transacties per dag uit te voeren, wat ook verklaard waarom deze systemen gebruikt worden door 71% van de Fortune 500 bedrijven wereldwijd. Deze machine heeft ook 40 terabyte werkgeheugen wat 1200 keer het aantal is in hedendaagse high-performance computers. (Tozzi, 2022)

De Z16 is ook 'Quantum safe': de bedreiging van Quantum computers in de toekomst blijft groeien en er is nog geen directe oplossing voor. IBM heeft hiervoor geïnvesteerd in Crypto Express 8S hardware security modules om data op de mainframe te beschermen en Quantum safe te maken. De nieuwe modules bevatten nieuwe quantum safe encryptie algoritmes die geëvalueerd zijn door de US National Institute of Standards and Technology. (Sayer, 2022)

Het is belangrijk om te weten dat de data niet wordt opgeslagen in de ASCII encoding maar in een binair formaat door middel van de EBCDIC encoding. (Singhal,

2023)
Dit is zo voor alle besturingssystemen die compatibel zijn op de mainframe behalve Linux for System z. Hierin is ASCII de standaard. (IBM, z.d.-b)

4.1.1. Het hoofdbesturingssysteem z/OS

Een IBM Mainframe ondersteund meerdere besturingssystemen maar de meest gebruikte is Z/OS. Dit is, samen met de IBM Z Systems, ontwikkeld door IBM waardoor dit het meest compatibel is met de hardware componenten en het blijft ondersteund door IBM zelf. Deze heeft verschillende karakteristieken zoals Workload Manager (WLM) om het uitvoeren van jobs in te plannen. Dit besturingssysteem kan gezien worden als hybride omdat het moderne taken van andere besturingssystemen neemt en combineerd met de architectuur van een IBM mainframe. Het heeft ook de mogelijkheid om terug te draaien naar een vorige versie zonder dat dit problemen zal veroorzaken met het systeem. (Rupp, 2022)

4.1.2. Software in z/OS

Dit besturingssysteem bevat tools zoals TSO, ISPF en SDSF om er een paar op te noemen. Via een 3270 terminal kan een gebruiker inloggen op de mainframe en gebruik maken van deze tools.

Time Sharing Option staat voor Time Sharing Option en is in principe de command line interface op de mainframe. Dit laat meerdere gebruikers toe om een interactieve sessie op te starten met z/OS via hun eigen inloggegevens. Zoals een traditionele CLI bestaat dit uit een prompt waar je commando's kunt ingeven om acties uit te voeren op het systeem. In Time Sharing Option wordt dit een 'READY' prompt genoemd omdat het een READY melding geeft als je commando's kunt invoeren. (IBM, z.d.-d)

Hoewel TSO vrij krachtig is, zal dit door de meeste eindgebruikers gebruikt worden in combinatie met ISPF of Interactive System Productivity Facility. Dit is een GUI dat bestaat uit menu's en panelen die allemaal verschillende functies kunnen uitvoeren (IBM, z.d.-d). Veel gebruikte functies zijn het aanmaken en schrijven van applicaties. ISPF biedt nog meer verschillende functies zoals het aanmaken van datasets tot een connectie maken met de database op de mainframe. Hierin kun je dus eigenlijk alles doen wat het platform te bieden heeft maar wel in de lijnen van de rechten die je hebt als gebruiker. (IBM, z.d.-d)

System Display and Search Facility of SDSF is volgens de IBM (2023) documentatie een interface om jobs en hun output te kunnen zien. Zo is er ook de mogelijkheid om een job te doen stoppen, bijhouden of vrijgeven. Met bijhouden wordt er bedoeld niet laten uitvoeren tot iemand een teken geef dat het uitgevoerd mag wor-

den. Vrijgeven wordt gebruikt om de fysieke middelen zoals de CPU vrij te geven zodat een andere job deze kan gebruiken.
Het biedt ook informatie over het z/OS systeem zodat u dit kan monitoren, managen en controleren. Hoewel het vooral gebruikt wordt om de status van een uitgevoerde job te zien, wordt deze tool ook gebruikt om fysieke toestellen te controleren (zoals een printer), de fysieke middelen zoals CPU of geheugen beheren en de system log en messages bekijken.

4.1.3. Programmeertalen in z/OS

Een mainframe systeem maakt vooral gebruik van Common Business-Oriented Language of COBOL. Dit lijkt sterk op het Engels en worrdt gebruikt om bedrijfgeoriënteerde applicaties te ontwikkelen voor het processen van commerciële data. Een COBOL programma wordt opgedeeld in 4 verschillende divisies (IBM, z.d.-a):

- Identification Division - Hier wordt info over het programma gegeven zoals de naam
- Environment Division - Hier komt de beschrijving van onderdelen van het programma die afhangen van de omgeving waarin het programma wordt uitgevoerd
- Data Division - Hier wordt de gebruikte data beschreven zoals input/output bestanden
- Procedure Division - Hier komt de effectieve logica van het programma

Naast COBOL is er ook nog Programming Language 1 of PL/1. Dit is geschikt voor commerciële en wetenschappelijke applicaties. Deze soort programma's bestaan uit "blocks" wat te vergelijken is met een groep van statements. Variabelen die in deze blocks worden gedeclareerd zijn enkel zichtbaar in die block of in een groep van blocks. Dit maakt PL/1 programma's zeer modulair. (IBM, z.d.-a)

4.1.4. Datasets

z/OS heeft ook een andere manier om data op te slaan door middel van datasets. Dit is volgens de documentatie van IBM (z.d.-d) een collectie van gerelateerde data records dat opgeslagen en opgehaald wordt door een toegewezen naam. Dit zou u kunnen zien als een bestand in andere besturingssystemen zoals in Windows of Linux.

Hier bestaan er verschillende soorten van:

- Sequentiële dataset
- Partitionele dataset (PDS)

- Virtual Storage Access Method (VSAM)

Sequentiële datasets bevatten records die achter elkaar opgeslagen zijn. Dit heeft als nadeel dat als een gebruiker bijvoorbeeld record 20 wilt lezen, moet hij eerst voorbij de voorgaande 19 records gaan. Deze soort datasets worden vooral gebruikt om grote hoeveelheden data in op te slaan. (IBM, z.d.-d)

Een partitionele dataset is meer voor programma's in te schrijven. Deze dataset bevat allemaal 'members' die op hun beurt dan de effectieve data bevatten. Dit heeft als voordeel dat je in een PDS de members kunt aanspreken in een willekeurige volgorde. Deze vorm van een dataset wordt ook wel een library genoemd. (IBM, z.d.-d)

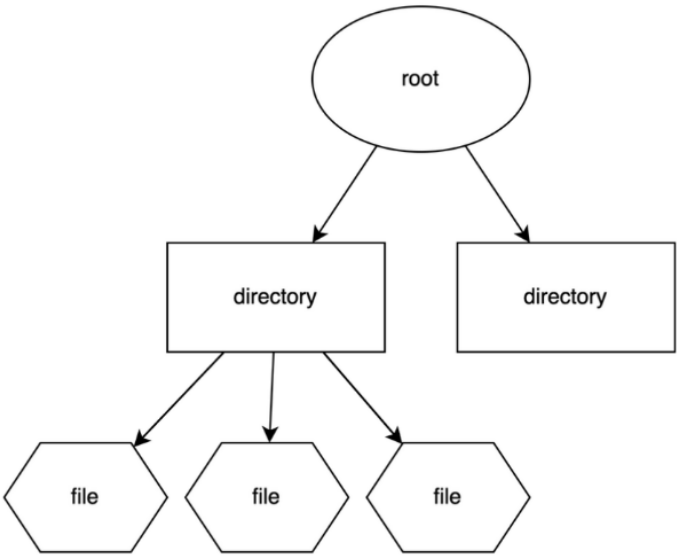
Een VSAM biedt een complexere manier van toegang tot verschillende soorten data en is vooral bedoeld voor applicaties. Door hun complexiteit kunnen ze niet frequent bekeken of aangepast worden zoals in ISPF. Een VSAM kun je verdelen in 4 verschillende datasets:

- Key Sequence Data Set (KSDS):
  Dit komt het meest voor en zal data opslaan op basis van een key, value systeem
- Entry Sequence Data Set (ESDS):
  Dit houdt de records in een sequentiële volgorde bij en worden ook gelezen in deze volgorde. Dit is vooral gebruikt door databanken en z/OS Unix bestanden.
- Relative Record Data Set (RRDS):
  Dit houdt records bij die je kunt ophalen op basis van een nummer. Zo heb je toegang tot records die opgeslagen zijn op plaats 100 zonder dat je door de eerste 99 records moet gaan. Dit is te vergelijken met een KSDS.
- Lineair Data Set (LDS):
  Dit houdt data bij in een byte stream en is de enige vorm van dit soort in een traditionele z/OS file.

(IBM, z.d.-d)

4.1.5. Unix op een mainframe

IBM heeft veel ingezet op modernisatie. Zo is er een Unix System Services (USS) omgeving bijgekomen die geïntegreerd is in het traditionele besturingssysteem z/OS. Er is ook een volledige mainframe die enkel Linux heeft als besturingssysteem namelijk de LinuxOne. In dit onderzoek zal er enkel gekeken worden naar een mainframe met z/OS die een USS omgeving heeft.



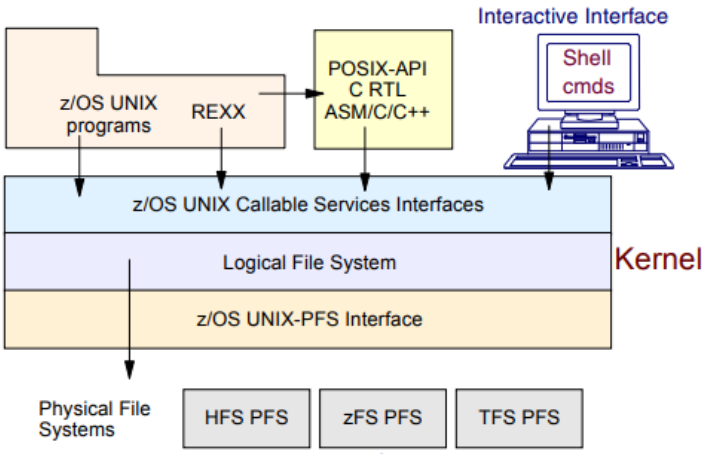
Figuur (4.1)
HFS voorbeeld (Codecadamy, 2022)

Omdat de USS dus samenwerkt met z/OS heb je veel meer functies die je kunt gebruiken. Zo is er de mogelijkheid op XML parsing, OpenSSH, de IBM HTTP Server for z/OS, de z/OS SDK for Java en nog veel meer. (Dhawan, 2013)

Dit systeem biedt een hierarchisch bestandssysteem (HFS) samen met een zSeries bestandssysteem (zFS). (Precisely, 2020)
De HFS is wel bekend voor de meeste UNIX gebruikers: Dit is een hierarchie van directories met bestanden of subdirectories die grafisch weergegeven kunnen worden in een tree view (HCL Technologies, 2022).

Zoals afgebeeld in figuur 4.1, is er een root (in Unix meestal afgebeeld als '/') die directories bevat. Deze directories bevatten bestanden en andere directories. Ze kunnen ook andere informatie bevatten over deze bestanden. Directories die opgeslagen zijn in een andere directory worden subdirectories genoemd. Vaak wordt er ook gebruik gemaakt van een parent-child verwijzing waar de parent directory een level boven de child of subdirectory zit. (Codecadamy, 2022)

zFS is iets minder bekend en kan gebruikt worden in plaats van of als toevoeging op het traditionele HFS. Dit heeft vooral zijn waarde door zijn sterke performantie in bestanden die vaak worden gebruikt. Het verminderd ook het risico van het verlies in updates omdat het data asynchroon schrijft in plaats van te wachten op een sync interval. Bestanden in dit systeem kunnen aangepast worden door middel van een Application Programming Interface (API) en kunnen zelf in de HFS toegevoegd worden zonder enige problemen. (IBM, 2012)



Figuur (4.2)
zFS werking (IBM, 2012)

In figuur 4.2 worden de bestanden opgehaald vanuit het zFS bestandsysteem door middel van de interactieve interface. Doankzij de API's tussen het logische -en fysieke bestandsysteem worden de juiste bestanden opgehaald en teruggegeven aan de gebruiker.

Unix bestanden op de mainframe worden bijna op dezelfde manier gebruikt als op een traditioneel unix systeem. Het kan een Java, C++ of Python programma bevatten. Deze programma's kunnen ook bestanden lezen of schrijven in een JSON of YAML formaat. Die kunnen op hun beurt dan gebruikt worden om analyses te doen op bepaalde data. Het hangt dus allemaal af van de use case om te zien op welke manier deze unix bestanden het best gebruikt worden. (Precisely, 2020)

Deze omgeving werkt ook met omgevingsvariabelen die invloed hebben op hoe het systeem werkt. Veel van deze variablen worden opgezet bij het inloggen maar de gebruiker kan deze zelf aanpassen. (Henry-Stocker, 2017)

4.1.6. Python AI toolkit for z/OS

Omdat we een Python interpreter zullen opzetten, zal er hier vooral gefocust worden op Python in z/OS. In Python wordt er vaak gebruik gemaakt van externe packages die publiek beschikbaar zijn. Deze moeten wel geïnstalleerd zijn op het systeem om er gebruik van te maken en dit kan een probleem zijn in USS. Door de beveiliging in een mainframe is het niet altijd even makkelijk om packages of andere direct te installeren in z/OS (IBM, 2021a) en hierdoor heeft IBM een Python AI toolkit for z/OS opgezet.

In de aankondiging door Evan Rivera (2023) werd deze toolkit beschreven als een bekende en flexibele ervaring voor developers tijdens het ontwerpen van AI oplos-

singen. Het biedt open source software en deze kunnen eenvoudig geïnstalleerd worden door de Package Installer for Python (pip).

De Python AI toolkit is dus een bibliotheek met verschillende, wereldwijd gebruikte Python packages voor AI en Machine learning workloads bijvoorbeeld NumPy, SciPy, Jupyter, etc. Elk van deze packages is volledig onderzocht geweest voor potentiële problemen in het systeem waardoor alles in deze toolkit voldoet aan de beveiligingsvoorwaarden zoals andere z/OS producten. (Bostian & Rivera, 2023)

4.1.7. Andere besturingssystemen

Zoals eerder vermeld is z/OS niet het enige besturingssysteem dat beschikbaar is op de mainframe. Hoewel dit door IBM aangeboden wordt, zijn er andere mogelijkheden die elk hun eigen sterktes en zwaktes hebben.

- z/Virtual Machine (z/VM)
  Dit is een type 1 hypervisor en kan gebruikt worden om meerdere besturingssystemen in te hosten. Dit bestaat uit een control program (cp) en een conversation monitoring system (CMS). De cp is verantwoordelijk voor het creëren van meerdere virtuele machines op basis van de fysieke hardware middelen. Het zorgt ook voor data en applicatie beveiliging voor alle systemen die in het systeem zitten. De CMS zit in een eigen virtuele machine en biedt een interactieve sessie tussen de andere virtuele machines en eindgebruikers. (IBM, z.d.-b)

- z/Virual Storage Extended (z/VSE)
  Dit besturingssysteem is vooral nuttig voor kleinere bedrijven die geen complexe batch -of transactie jobs moeten processen. Het is mogelijk dat naarmate ze groeien, ze overgaan naar z/OS als z/VSE niet genoeg blijkt te zijn. Het design van dit besturingssysteem maakt het perfect voor meer routine batch jobs parallel uit te voeren. Meestal wordt z/VM ook gebruikt als een terminal interface voor development en systeembeheer in z/VSE. (IBM, z.d.-b)

- Linux for System z
  Dit is zoals de naam zegt, Linux op de mainframe. Er zijn hier 2 verschillende distributies voor: Linux for S/390 (gebruikt 31-bit adressering) en Linux for System z (gebruikt 64-bit adressering)
  Linux for System Z refereerd naar Linux for S/390. Dit besturingssysteem maakt niet gebruik van een 3270 terminal maar X-Windows based terminals. Dit bestaat uit een cli waarmee je meestal een telnet of ssh connectie maakt met het Linux for System z besturingssysteem. Dit is ook volledig in de encoding

ASCII en niet EBCDIC. (IBM, z.d.-b)

- z/Transaction Processing Facility (z/TPF)
  Dit besturingssysteem is vooral nuttig voor bedrijven die veel transacties moeten uitvoeren zoals een bank of vliegmaatschappij. Dit kan gebruik maken van verschillende mainframes om tienduizende transacties per seconde uit te voeren zonder enige onderbreking. (IBM, z.d.-b)

4.2. IBM Developer for z/OS

4.2.1. IDz

De testomgeving waarin we zullen werken is IBM Developer for z/OS (IDz) versie 16.0.2. Dit programma is volgens de definitie van Spohn (2023) een toolset voor het ontwikkelen en opzetten van hybride cloud applicaties op z/OS.
Het is een Eclipse based programma met de mogelijkheid om een connectie te maken met verschillende omgevingen op de mainframe (bv de USS of z/OS omgeving). Zo kunt u alle bestanden zien, openen en aanpassen. Omdat de data nogsteeds op de mainframe staat, kunnen wijzigingen direct gezien worden ookal bekijkt u het in een ander programma. Als u bijvoorbeeld een bestand wijzigt via IDz, kunt u de wijzigingen direct zien in ISPF.
Dit programma is vooral ontwikkelt om een envoudige IDE te bieden om in te programmeren aangezien niet iedereen bekend is met ISPF.

Zoals vermeld zal er gebruik worden gemaakt van IDz versie 16.0.2 . In het overzicht van IBM (2024b), ondersteund deze versie syntax veranderingen voor COBOL 6.4, PL/1 6.1 en REXX. Er is ook een ZUnit update wat vooral het gebruik van deze tool makkelijker maakt.

ZUnit staat voor z/OS Automated Unit Testing en is een framework dat gebruikt wordt in IDz om COBOL en PL/1 programma's te testen. Hiervoor maakt het gebruik van verschillende 'samples' die door IBM ontworpen zijn (bv. Enterprise COBOL CALL02.cbl sample test case). Het gebruik van deze tool maakt het makkelijker voor developers om hun code (geschreven in COBOL of Pl/1) te testen op de mainframe. Dit maakt het schrijven van code in IDz veel efficiënter. (IBM, 2024a)

4.2.2. Eclipse IDE

zoals eerder vermeld is IDz gebaseerd op de Eclipse Integrated Development Environment (IDE) wat ontwikkeld is door de Eclipse Foundation in 2001. Het is begonnen sinds IBM 3 miljoen lijnen code van hun Java tools heeft gedoneerd om een open source IDE te creëren. Het is dus Java gebaseerd maar is kan nogsteeds gebruikt worden voor andere programeertalen zoals Python door de verschillende

plug-ins die beschikbaar zijn gemaakt door de community achter Eclipse. (Hanna, 2021)

Eclipse zelf heeft ook de mogelijkheid om verder uit te breiden door plug-ins. Dit zijn toevoegingen aan de gebruikte software die het mogelijk maken om applicaties, computer programma's en web browsers aan te passen. Dit zijn ook add ons die extra functionaliteiten kunnen bieden aan het gebruikte programma. (George, 2021)

4.2.3. Wizards

Een programma zoals IDz heeft veel mogelijkheden om instellingen aan te passen door middel van wizards. Dit is een serie van pagina's dat de gebruiker begeleidt om een complexe taak uit te voeren. Elke pagina vraagt wat informatie en als de gebruiker op 'finish' klikt wordt de taak uitgevoerd. Er is ook altijd een mogelijkheid om het proces stop te zetten. (Eclipse, 2006)

4.3. IBM Z in een bankomgeving

Dit onderzoek wordt uitgevoerd in een bankomgeving dus is het wel interessant om na te gaan welke voordelen deze technologie te bieden heeft in deze omgeving.

Volgens Turner (2022) gebruiken de meeste banken een IBM mainframe omdat ze de rekenkracht kunnen bieden die banken nodig hebben om efficiënt te kunnen werken. Kenmerken zoals robuustheid, betrouwbaarheid en snelle processing kracht spelen ook een grote rol aangezien het van groot belang is dat het systeem bijna altijd actief moeten zijn. De mainframe toont hier zijn sterkte door de 8 nines oftwel 99,999999% van de tijd beschikbaar per jaar. (IBM, z.d.-c)

Dit is niet de enige reden waarom het gebruikt wordt door 92 van de top 100 banken ter wereld (Tozzi, 2022). In een aankondiging van IBM (2022) over hun nieuw systeem de IBM z16, werd er gezegd dat 70% van wereldwijde transacties door een mainframe verwerkt worden.

Beveiliging speelt ook een grote rol, een studie van Morning Consult en IBM (2022) toont aan dat krediet kaart fraude het meest voorkomende type fraude is onder klanten in 7 verschillende landen waaronder Duitsland, de Verenigde Staten en China. de IBM zSystems heeft al een goede transactie beveiliging zonder teveel vertraging maar de z16 brengt hier nog een AI model bij. Hierdoor kunnen banken transacties controleren op een veel grotere schaal: 300 miljard door AI beveiligde transacties per dag met maar 1 milliseconde vertraging. Andere zaken zoals belastingsfraude kunnen ook vermeden worden hiermee. (IBM, 2022)

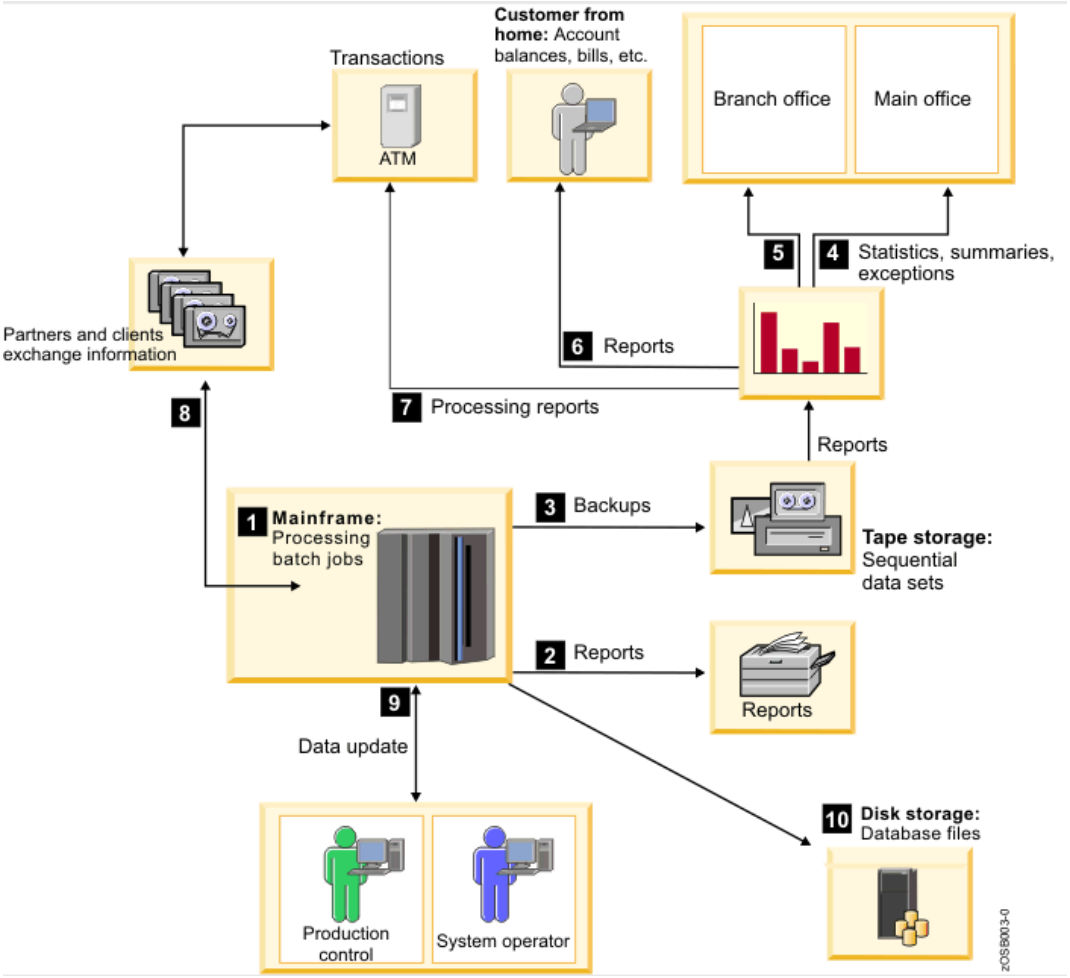
IBM doet dit door middel van een nieuwe Telum Processor. Het is ontworpen om deep learning algoritmes naar workloads te brengen voor bedrijven om in real-time fraude te detecteren en te verwerpen. Dit is de eerste processor van IBM dat gebruik maakt van AI terwijl de transactie wordt uitgevoerd. Naast het detecteren van fraude is deze chip nog nuttig voor loan processing, het tegengaan van witwassen van geld en risico analyse. (IBM, 2021b)

4.3.1. Batch & online jobs

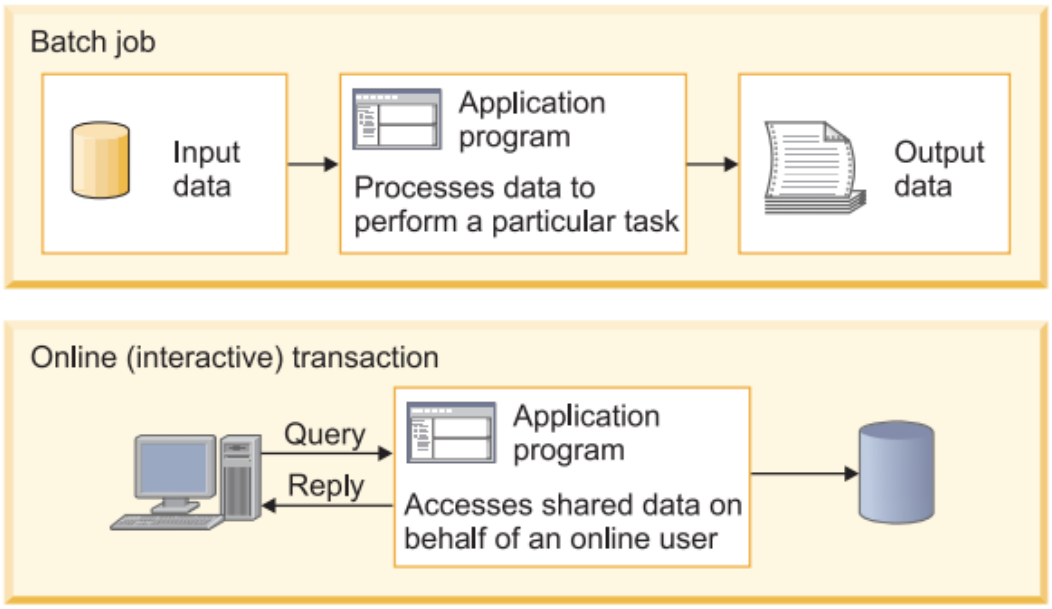
Batch processen is ook een belangrijk voordeel dat een mainframe te bieden heeft. Dit zijn jobs die uitgevoerd kunnen worden met weinig of geen interactie van een gebruiker (IBM, z.d.-d). Dit is vooral nuttig voor taken die op vaste momenten moeten uitgevoerd worden.
In figuur 4.3 is een duidelijk voorbeel over hoe een batch job te werk gaat:

1. De batch job wordt opgestart door de mainframe
2. De job genereert statistieken van het bedrijf
3. Er wordt een backup gemaakt van de kritische bestanden en de database voor en na de job
4. De statistieken worden verstuurd naar een specifieke plaats zodat het geanalyseerd kan worden
5. Fouten in de statistieken worden in een andere locatie geplaatst
6. Informatie van klanten hun bankrekening wordt gemaakt en verstuurd.
7. Een verslag van de processing tijd wordt verstuurd naar de partners van de bank
8. De partner ontvangt het verslag
9. Het systeem monitort de uitkomst van de batch job
10. Jobs en transacties updaten de database zodat de progressie wordt opgeslagen

Batch jobs zijn niet de enige vorm van jobs die uitgevoerd worden op een mainframe. Online transaction processing of OLTP is zeer belangrijk in een bank. In contrast met batch processing, heeft dit een eindgebruiker nodig die de job opstart. Het heeft ook geen vast moment waarop de job uitgevoerd wordt en kan dus op elk moment van de dag. Meestal duren deze transacties niet lang maar de systemen die verantwoordelijk zijn om deze jobs uit te voeren moeten veel verschillende gebruikers op hetzelfde moment ondersteunen zonder enige vertraging. (IBM, z.d.-b)



Figuur (4.3)
Batch Job (IBM, z.d.-b)



Figuur (4.4)
Grafisch verschil tussen batch en OLTP (IBM, z.d.-b)

Een grafische voorstelling van de verschillen tussen een batch job en een online transaction job vind u in figuur 4.4. Een batch job heeft dus input die verwerkt wordt door een programma op een bepaald, vast moment. De uitkomst wordt dan geschreven in een bestand om analyses op uit te voeren.
Een online transactie daarentegen wacht op een query van een gebruiker om te starten. Data wordt dan opgehaald uit het systeem en teruggegeven aan de gebruiker. (IBM, z.d.-b)

4.3.2. Uitvoeren van batch jobs

Batch jobs worden uitgevoerd door middel van Job Control Language (JCL). Dit is een programma dat de middelen definiëerd die de batch job nodig heeft bijvoorbeeld een input -of output bestand. Een JCL programma bestaat uit 3 statements:

- JOB - Hier wordt de naam van het programma bepaald (jobname). Er kunnen parameters toegevoegd worden die gelden voor de hele job.
- EXEC - dit bepaald het programma die uitgevoerd moet worden. Een JCL kan meerdere EXEC statements bevatten waarbij elke statement een *job step* wordt genoemd
- DD - Dit bepaald de input en/of output bestanden die de batch job nodig heeft.

Dit kan zowel in TSO als in ISPF uitgevoerd worden. (IBM, z.d.-d)

In DNB wordt de JCL opgesplitst in 2 programma's:

1. Job card - Bevat de JOB statement
2. Procedure - Bevat de EXEC en DD statements

Voor het uitvoeren wordt er enkel verwezen naar de Job card en die roept op zijn beurt de procedure op.

5

Opzetten Pydev

Het programma waarin het onderzoek uitgevoerd zal worden is IBM Developer for z/OS (IDz) versie 16.0.2 met voorgeïnstalleerde programma's van DNB zelf. IDz is gebaseerd op Eclipse versie 4.23.0 .
In dit onderzoek zullen we versie 8.2.0 van PyDev installeren. Dit is een iets oudere versie maar het best compatibel met IDz, de andere versies kunnen problemen met zich meebrengen tijdens en na de installatie. Versie 8.2.0 bevat onder andere een debugger, een code formatter en code analyse. Het is belangrijk om te weten dat IDz door IBM beschikbaar wordt gesteld en ze dus support zullen bezorgen waar nodig moesten er problemen zijn met hun programma. Dit is niet het geval met PyDev omdat dit door een externe partij is ontwikkeld en dus niet ondersteund wordt door IBM.

5.1. Vereisten voor installatie

- Eclipse versie 4.6 based programma nodig - IDz
- Java 11 of hoger
- Python 2.6 of hoger

5.2. Installatie

Om Pydev te gebruiken moeten we het eerst installeren via hun Github repository https://github.com/fabioz/Pydev/releases.
Dit zal een .zip installeren die je moet uitpakken en ergens moet opslaan. Het uitpakken kan een enige tijd duren.

In IDz navigeer je naar *help -> install New Software*

Je zult een installatie wizard te zien krijgen en hierin klik je op *add -> local*

Selecteer de map waar Pydev in is uitgepakt. klik op *add*

Je krijgt weer een scherm te zien met de opties die je kunt installeren. Het kan zijn dat het eerst leeg is en dit verander je door in de checkbox de Group items by category uit te vinken. Hierna krijg je 3 opties.

- PyDev for Eclipse 8.2.0.202102211157
- PyDev for Eclipse Developer Resources 8.2.0.202102211157
- PyDev Mylyn Integration 0.6.0

De eerste is noodzakelijk en de tweede optioneel. De derde mag niet aangevinkt worden aangezien MyLyn niet geïnstalleerd is in IDz waardoor het opzetten van PyDev zal falen.

Klik op *next*, dit zal alles opzetten en kan even duren.

Als dit gedaan is wordt er een overzicht gegeven van de installatie details. Hier klik je op *finish*.

In de balk rechtsonder wordt de status van de installatie weergegeven.

Als het klaar is met installeren zal het een pop-up geven om IDz herop te starten. Als u nog open projecten heeft is het best om deze eerst op te slaan en dan zelf te herstarten. Anders klikt u op *Restart Now*

Om te zien of de installatie gelukt is navigeert u naar *help -> about -> Installation details*

Geef in de zoekbalk “Pydev”. Hier zou u de geïnstalleerde software moeten zien

Als je een Python script opent zal het nogsteeds gebeuren in de interne text editor. Om dit te wijzigen moeten we de file associations bekijken. Deze zullen bepalen welke editor er gebruikt wordt bij een bepaalde extensie. Hier zullen we dus de Python editor van Pydev linken aan de .py extensie.

Ga naar *Window -> Preferences -> General -> Editors -> File Associations*

Hier is een lijst met allemaal extensies en het programma waarmee ze geopend worden. IDz heeft niet standaard een python editor dus deze moet toegevoegd worden. Klik bij “file types” op *add*

Geef hier .py in en klik op ok

In de lijst van extensies en ziet hier normaal .py tussenstaan

Selecteer de py extensie en bij “associated editors” komen er 3 opties:

- Python Editor
- Text Editor
- Generic Text Editor

Selecteer “Python Editor” en klik op *Default*.

Als je een python script opent zal dit automatisch gebeuren in de python editor en zijn alle functies van Pydev beschikbaar.

6

Runtime environment opzetten

6.1. Uitvoeren van Python scripts op een lokale machine via IDz

In IDz is er de mogelijkheid om programma's uit te voeren op de lokale machine waarop IDz uitgevoerd wordt. Hiervoor is er een Python executable nodig op deze machine. Deze moet geïnstalleerd zijn en in IDz moet er een vewijzing naar gedaan worden.

DNB heeft strenge regels op het installeren van externe software op laptops die een connectie kunnen maken met de mainframe. Door deze veiligheidsredenen is er geen mogelijkheid om Python te installeren en kan deze stap ook niet uitgevoerd worden.

Moest dit kunnen kan er naar de Python executable verwezen worden in IDz via *Window -> preferences -> Pydev -> Interpreters -> Python Interpreter*.
Hier klik je op *New...* en dan verschijnt er een wizard om de interpreter te kiezen. Zoek de geïnstalleerde Python interpreter en klik op *Apply*.

Als je rechtsklikt op een Python bestand en naar *Run* navigeert, zie je de Python interpreter staan.
Vermoedelijk zal dit het Python script uitvoeren op de gebruikte machine en niet in de USS omgeving op de mainframe.

6.2. Uitvoeren van Python scripts in de USS omgeving via IDz

De Python interpreter is niet nodig op de lokale machine voor het uitvoeren van Python programma's in de USS omgeving. Hiervoor moeten we de remote systems explorer perspectief openen via *Window -> Perspective -> Open Perspective -> Other -> Remote System Explorer*.

Maak een connectie met de USS omgeving met de juiste inloggegevens en navigeer naar het tablad *Remote Shell*.

Deze zal leeg zijn maar door te klikken op de 3 puntjes met als naam *View Menu* kunnen we de connectie toevoegen. Klik op *Launch* en kies de juiste connectie. Dit zou 2 connecties moeten tonen. De eerste is de connectie met de lokale machine en de tweede is die met de USS omgeving. Als je de connectie met de USS omgeving kiest zal het 2 keuzes geven:

1 z/OS UNIX Shells

2 TSO Commands

Om met de USS omgeving connectie te maken kiezen we optie 1: *z/OS UNIX Shells*. De tweede optie is om TSO commando's te geven in de z/OS omgeving waarmee we verbonden zijn.

Nu zal er een kleine terminal opstarten waar we zijn ingelogd op de mainframe in de USS omgeving. Hierin kunnen we dus navigeren naar verschillende directories met Python bestanden in en uitvoeren met het Python commando. We kunnen ook directories of bestanden aanmaken in deze terminal zonder een externe ssh connectie nodig te hebben.

Belangrijk om te vermelden is dat de Python installer moet zijn gedefinieerd anders kun je geen Python bestanden uitvoeren in USS. Dit is een globale variabele die je moet initializeren.

7

Testen Runtime environment

In dit hoofdstuk zal er onderzocht worden hoe krachtig de ingebouwde terminal van IDz is wat zijn limitaties zijn. Dit doe ik voor hoofdstuk 8 omdat er hier de omgeving wordt opgegezet om de het testscript in uit te voeren.

7.1. Opzetten testomgeving

Alle bestanden die worden geschreven om te testen zullen aangemaakt worden in de directory */tmp/BPtestfolder*. Deze wordt aangemaakt via het Bash commando:

```
$ mkdir /tmp/BPtestfolder
$ cd /tmp/BPtestfolder
```

Er wordt verwacht dat Python correct is geïnstalleerd in de USS omgeving.

7.2. Test script aanmaken en uitvoeren

Om het Python bestand op te slaan, zal er een directory gemaakt worden genaamd */tmp/BPtestfolder*. Hierin wordt er een testt.py bestand aangemaakt die *Hello World* zal afdrukken op het scherm. Aangezien dit geen groot programma is, zal dit met het *echo* commando geschreven worden in het Python bestand als volgt:

```
$ mkdir /tmp/BPtestfolder
$ touch test.py
$ echo `print('Hello World')` >> test.py
$ python test.py
```

Dit programma geeft *Hello World* terug en werkt dus perfect.

7.3. Opzetten Python virtuele omgeving

Een Python virtuele omgeving wordt vaak gebruikt om toegangsrechten te ontwijken en een eigen omgeving opzetten waar de gebruiker toegang heeft tot alle functies zoals bijvoorbeeld packages installeren. Deze virtuele omgeving zullen we proberen opzetten en activeren in IDz. Als dit lukt zullen we packages installeren en testen of we deze kunnen gebruiken.

```
$ python -m venv --system-site-packages ./virtual_env
$ cd virtual_env/bin
$ activate
$ pip list
```

Hoewel de virtuele omgeving aangemaakt wordt, kan deze niet opgestart worden met het source commando wat wel gaat in een ssh connectie. In IDz moeten we naar de *bin* directory gaan en het *activate* programma uitvoeren. We moeten in de *bin* directory blijven om te kunnen werken in de virtuele omgeving. Het commando *pip list* wordt gebruikt om te testen of de virtuele omgeving actief is. Moest het niet actief zijn, zou dit commando niet lukken. Hierin zien we een lijst met geïnstalleerde packages die op het root systeem geïnstalleerd zijn. Dit hebben we gedaan door de *–system-site-packages* optie bij het aanmaken van de virtuele omgeving.

7.4. Installeren van packages

In deze virtuele omgeving zijn er nog packages nodig waar Python programma's gebruik van kunnen maken. Deze zullen we halen uit de Python AI toolkit for z/OS van IBM of van de Pypi website. De packages die geïnstalleerd zullen worden, zijn nodig voor het testprogramma die wordt geschreven in hoofdstuk 8. Deze packages zijn *FastApi*, *Uvicorn* en *Jsonschema*.

FastAPI en Uvicorn maken geen onderdeel uit van de Python AI toolkit dus zal er een .whl file geïnstalleerd moeten worden van https://pypi.org/. Het is belangrijk om alle packages waarvan FastAPI gebruik maakt, ook geïnstalleerd moeten zijn. Via drag and drop is het mogelijk om het .whl installatiebestand in de USS omgeving te krijgen vanaf de lokale machine. Dit moet in 'binary' gebeuren omdat het anders foutmeldingen zal geven tijdens het installeren. Dit kan ingesteld worden in IDz via *Windoz -> Preferences -> Remote Systems -> Files*. Hier wordt een lijst van extensies weergegeven en hoe ze worden overgezet naar de USS omgeving. Om een .whl bestand correct over te zetten naar de mainframe klik je op *Add...* en geef je .whl in. Dit bestandstype komt in de lijst terecht en als dit geselecteerd is, moet de *Default File Transfer Mode* op *Binary* staan.

Jsonschema is beschikbaar via de Python AI toolkit en staan dus al op de main-

frame. Dit zijn ook .whl bestanden.

Er zal ook gebruik worden gemaakt van de ZOAU maar deze zijn al geïnstalleerd op het systeem en hebben geen verdere configuratie nodig.

Alle nodige packages zullen opgeslagen worden in de directory */tmp/BPtestfolder/-virtual_env/packages*. Via het *cp* commando worden de packages uit de Python AI toolkit gekopiëerd naar de aangemaakte directory:

```
$ mkdir /tmp/BPtestfolder/virtual_env/packages
$ cp /Path/To/AI/Toolkit/jsonschema-4.17.3-py3-none-any.whl \
                                    /tmp/BPtestfolder/virtual_env/packages
```

Al deze packages moeten apart geïnstalleerd worden via het *pip install* commando. Aangezien deze bestanden op het systeem staan en we ze niet via het internet willen installeren, wordt de optie *–no-index* toegevoegd. Dit laat het systeem weten dat we het niet online willen installeren. De installatie van de pypi packages gebeurd als volgt:

```
$ pip install ../packages/annotated_types-0.4.0-py3-none-any.whl --no-index
$ pip install ../packages/typing_extensions-4.10.0-py3-none-any.whl --no-index
$ pip install ../packages/pydantic-1.10.15-py3-none-any.whl --no-index
$ pip install ../packages/idna-3.6-py3-none-any.whl --no-index
$ pip install ../packages/sniffio-1.3.1-py3-none-any.whl --no-index
$ pip install ../packages/anyio-4.3.0-py3-none-any.whl --no-index
$ pip install ../packages/starlette-0.35.1-py3-none-any.whl --no-index
$ pip install ../packages/fastapi-0.109.0-py3-none-any.whl --no-index

$ pip install ../packages/h11-0.14.0-py3-none-any.whl --no-index
$ pip install ../packages/uvicorn-0.25.0-py3-none-any.whl --no-index

$ pip install ../packages/attrs-23.2.0-py3-none-any.whl --no-index
$ pip install ../packages/pyrsistent-0.20.0-py3-none-any.whl --no-index
$ pip install ../packages/jsonschema-4.17.3-py3-none-any.whl --no-index
```

7.5. Omgevingsvariabelen instellen

Zoals eerder vermeld, zal er gebruik worden gemaakt van ZOAU om datasets te lezen en schrijven. Deze package is al geïnstalleerd maar er zullen nog omgevingsvariabelen ingesteld moeten worden zodat dit correct functioneert. De omgeveringsvariabelen die gewijzigd moeten worden zijn *ZOAU_HOME*, *PATH* en *LIBPATH*.

Hiervoor gebruiken we het *export* commando als volgt:

```
$ export ZOAU_HOME=/pp/idz/v16r0/zoautil/
$ export PATH=$ZOAU_HOME/bin:$PATH
$ export LIBPATH=$ZOAU_HOME/lib:$LIBPATH
```

Moesten deze variabelen niet gewijzigd zijn, zou het volgende foutmelding geven:

```
CEE3201S The system detected an operation exception
                                        (System Completion Code=0C1).
From entry point _zoau_io_zopen at compile unit offset +0000000029B242B4
                at entry offset +00000000000002D4 at address 0000000029B242B4.
Killed
```

8

Testen Pydev

Om Pydev te testen zullen we een FastAPI schrijven in IDz. Deze API heeft als functie om van een batch job, de 2 JCL programma's aan elkaar te binden en te schrijven in een PDS member. Hierdoor testen we het gebruik van de packages die geïnstalleerd zijn, alsook de verbinding met z/OS.

Het doel van dit hoofdstuk blijft nog steeds de ervaring testen van een volledig Python programma schrijven door middel van Pydev en zien hoe efficiënt dit eigenlijk is. Hier zal er ook gezorgd worden dat deze API gebruik maakt van de ZOAU om datasets te lezen en schrijven. Er is ook een kleine JSON file aanwezig die de naam van de header -en procedure JCL bevat.

Door deze verschillende technologieën te gebruiken testen we niet enkel hoe complexe programma's err geschreven kunnen worden in Pydev, maar ook hoe efficiënt het is om op de mainframe gebruik te maken van deze moderne technieken in combinatie met de oudere.

8.1. Schrijven van een test API

Verder staat de test API die geschreven werd. Het bevat 1 functie die 2 JCL's uit het JSON bestand van een meegegeven job ophaald en aan elkaar schrijft in een dataset die gedefinieerd is in de variabele *dsMemberPath*.

```
from fastapi import FastAPI
import json
from zoautil_py import zsystem, datasets
from zoautil_py.zoau_io import zopen

app = FastAPI()
```

```
@app.get("/api/v1/{environment}/proc/{proc}")
def get_jcl(environment: str, proc: str):
    # ========================================================== #
    # ================= Declaring variables ================== #
    # ========================================================== #

    # uppercase the given environment variable
    environment = environment.upper()

    # header_path = path of the jobcard
    # body_path = path of the procedure
    header_path, body_path = get_values(environment)

    # List to store the lines in of both the job card and procedure
    records = []

    # Output dataset
    outds = "AD12215.API.OUT" # Dataset
    member = proc # Dataset member
    dsMemberPath = f"{outds}({member})" # Full path of the dataset member


    # ========================================================== #
    # ============ Check if input datasets exist ============= #
    # ========================================================== #
    # Both files exist
    if (proc in datasets.list_members(header_path)) \
                        & (proc in datasets.list_members(body_path)):
        print(f"Both members found ({member}), \
            returning both the job card and procedure")

        # Reading and appending JOB file
        with zopen(f"//'{header_path}({proc})'", 'r', "cp1047") \
                        as record_stream:
            for record in record_stream:
                records.append(record.upper())

        # Reading and appending PROC file
        with zopen(f"//'{body_path}({proc})'", 'r', "cp1047") \
                        as record_stream:
```

```
            for record in record_stream:
                records.append(record.upper())

    # Only header file exists
    elif (proc in datasets.list_members(header_path)) \
                        & (proc not in datasets.list_members(body_path)):
        print(f"Member {body_path}({proc}) not found in procedure path, \
                        returning only {header_path}({proc})")

        # Reading and appending JOB file
        with zopen(f"//'{header_path}({proc})'", 'r', "cp1047") \
                        as record_stream:
            for record in record_stream:
                records.append(record.upper())

    # Only body file exists
    elif (proc not in datasets.list_members(header_path)) \
                        & (proc in datasets.list_members(body_path)):
        print(f"Member {header_path}({proc}) not found in job card, \
                        returning only {body_path}({proc})")

        # Reading and appending PROC file
        with zopen(f"//'{body_path}({proc})'", 'r', "cp1047") \
                        as record_stream:
            for record in record_stream:
                records.append(record.upper())

    # No files exist
    else:
        return "files don't exist"

    # ============================================================ #
    # ============ Writing output into a dataset ============= #
    # ============================================================ #

    # Deleting older output written if it exists
    if member in datasets.list_members(outds):
        datasets.delete_members(f"{dsMemberPath}")
        if member in datasets.list_members(outds):
            return f"Dataset member {dsMemberPath} \
                        can't be deleted because it is open"
```

```
        # Writing the content of "records" into the dataset "dsMemberPath"
        return write_into_dataset(dsMemberPath, records)


def get_values(environment: str):

    # Opening the json file with the file paths
    with open("config.json", 'r', encoding="utf-8") as test_file:

        # Loading the json in the variable json_data
        json_data = json.load(test_file)

        # Assign the return values with the correct filepaths
        header_path = json_data[environment]["HEADER"]
        body_path = json_data[environment]["BODY"]
        return header_path, body_path


def write_into_dataset(dsname: str, records: list):
    for record in records:
        # Remove all control characters for the record
        mapping = dict.fromkeys(range(32))
        res = record.translate(mapping)
        # Writing record into dataset
        datasets.write(dsname, res, append=True)

    return f"Output written in PDS {dsname}"
```

Om de datasets te lezen, wordt er gebruik gemaakt van het *zopen* statement van ZOAU. Dit is het equivalent van het traditionele *open* statement in Python om een bestand te openen. Hier gebruiken we 3 positionele parameters:

1. Het te openen bestand. Dit bevat de naam van de dataset.
2. Dit bepaald wat er gebeurd wordt met het bestand. *'r'* voor te lezen en *'w'* om te schrijven
3. De encoding van het meegegeven bestand

Het is belangrijk om de encoding mee te geven bij het oproepen van deze functie. Datasets worden opgeslagen in de encoding *IBM-1047* en dit wordt gedefinieerd als *'cp1047'*

9

Testen van de volledige applicatie

Om de API uit te voeren zal er gebruikt worden van de Uvicorn package met het volgende commando:

```
$ python -m uvicorn main:app
```

De API zou hierdoor moeten luisteren op poort 8000 van het loopback adres van z/OS voor inkomende verzoeken.

10

Evaluatie test

Bij het uitvoeren van de API gaf dit een foutmelding in IDz maar niet in de ssh connectie. Er lijkst

11

Conclusie

Deze proof of concept biedt een meerwaarde aan de modernisatie van het mainframe systeem die er volgens mij zit aan te komen. Deze systemen zijn nog heel belangrijk maar werkt met te oude technieken waardoor er niet veel mensen zijn die ze kunnen onderhouden. Het toevoegen van modernere manieren van werken is niet voldoende, ze moeten ook efficiënt zijn om mee te kunnen werken wat niet het geval is in de standaard versie van IDz.

Het opstellen van deze plug-in is niet zeer complex maar er zijn een paar instellingen die niet voor de hand liggend zijn en voor problemen kunnen zorgen. Aangezien IDz ook veel verschillende functionaliteiten heeft weten veel mensen niet waar ze moeten zoeken als ze iets zoals dit willen opzetten.

Het is zeer makkelijk om alles in 1 IDE te hebben tijdens het programmeren omdat je niet telkens van programma naar programma moet springen. Persoonlijk vind ik de terminal in Idz om de Python scripts in uit te voeren niet veel beter om in te werken en verkies zelf liever een ssh connectie via powershell. Tijdens het testen heb ik gemerkt dat iets complexere zaken niet direct lukken zoals een Python virtuele omgeving opzetten. In de USS omgeving maakt dit het moeilijk om packages te installeren die gebruikt worden in bijna alle Python programma's.

Zelf had ik niet verwacht dat het mogelijk was om Python programma's uit te voeren in USS via IDz en was deels verbaasd toen ik ondervond dat het wel kon. Dit toont voor mij nog eens aan hoe groot Eclipse based programma's zoals IDz kunnen zijn.

A

Onderzoeksvoorstel

Het onderwerp van deze bachelorproef is gebaseerd op een onderzoeksvoorstel dat vooraf werd beoordeeld door de promotor. Dat voorstel is opgenomen in deze bijlage.

Samenvatting

IBM heeft voor zijn IBM Z Systems de deur opengezet naar vernieuwing door Java en Python toegankelijk te maken op de Mainframe. Hoewel deze talen ondersteund worden, is het niet altijd even makkelijk om programma's hierin te schrijven. IBM Developer for z/OS is een stap in de goede richting door zijn functionaliteit om in een bekend programma scripts te openen en aan te passen die opgeslagen zijn op mainframe. Aangezien dit Eclipse based is, is er geen Python interpreter waardoor Python scripts aangepast moeten worden in een text editor.

Deze paper zal een proof of concept zijn voor het opstellen van Pydev en een Python IDE in IBM Developer for z/OS. Hierdoor kunnen Python developers efficiënter code schrijven, aanpassen en submitten in de Unix System Services omgeving van de mainframe. Verder zal er ook nog onderzocht worden om de output van de Python programma's te zien in de console van IBM Developer for z/OS. Dit is belangrijk voor de programmeurs van DNB die werken in dit programma die nu hun Python code moeten testen in een ssh connectie.

In dit onderzoek zal er gewerkt worden met de Pydev plug-in voor Eclipse en zal er een stap voor stap instructie gegeven worden over hoe dit ingesteld moet worden. Ook zal er onderzocht worden of deze scripts uitgevoerd kunnen worden door IBM Developer for z/OS in de Mainframe omgeving.

Het resultaat zal een volledige Proof of concept zijn voor een Python interpreter op te zetten in dit programma. Dit is vooral relevant voor Python developers in bedrijven die IBM Developer for z/OS gebruiken in combinatie met een IBM Z Mainframe.

A.1. Introductie

In de huidige technologische wereld is het moeilijk bij te houden welke mogelijkheden er zijn om bepaalde projecten uit te voeren. Zo heb je altijd nieuwe technieken die net iets efficiënter of krachtiger zijn dan een ander. Door deze voortgaande evolutie zullen individuen niet direct op de hoogte zijn van recente ontwikkelingen, maar vergeten ze ook de oudere toepassingen van een bepaalde techniek. Dit is vooral merkbaar bij IBM hun z Systems of mainframes die zo belangrijk, maar zo snel vergeten worden in de IT wereld. Hoewel het een zeer hoogwaardige technologie is, zijn de technieken nog steeds zeer oud. Hier wordt wel op ingezet door modernere talen zoals Java en Python compatibel te maken met dit systeem. Een veel gebruikte tool, aangeboden door IBM, is IBM Developer for z/OS. Dit is een Eclipse based programma geschreven in Java en kan een connectie maken met de mainframe en zijn opgeslagen bestanden. Dit maakt het eenvoudiger voor gebruikers om verschillende bestanden te openen maar heeft geen Python interpreter waardoor het voor developers niet zo efficiënt is om Python programma's aan te passen.

A.1.1. Probleemstelling

Het bedrijf waar het onderzoek in wordt uitgevoerd is Den Norske Bank (DNB) met een hoofdvestiging in Oslo, Noorwegen. Hun developers zoals Stian Botnevik zitten vast met het beschreven probleem en zoeken een efficiëntere manier om Python scripts te schrijven en testen. Momenteel passen ze Python scripts aan in een text editor van IBM Developer for z/OS (IDz) en testen ze de code in een ssh connectie. Python scripts worden opgeslagen en uitgevoerd in de Unix System Services (USS) omgeving. Dit is een Unix omgeving op de mainframe. In IDz kunnen developers alle bestanden openen in de USS omgeving wat duidelijker is dan code aanpassen in een terminal via bijvoorbeeld de vim tool.

A.2. State-of-the-art

A.2.1. Een andere manier van werken

Veel bedrijven die werken met een mainframe hebben het probleem dat ze onvoldoende mensen vinden met de nodige kennis over deze technologie. De introductie van Unix systemen op de mainframe in het jaar 2000 (Mertic, 2020) had dit probleem wat verminderd maar zeker niet weggewerkt. De oorzaak van deze situatie is door de oude technieken die het systeem gebruikt. COBOL en PL/1 zijn niet de meest aantrekkelijke programmeertalen om te leren en mensen kiezen liever Bash als scripttaal in plaats van REXX. Hoewel IBM veel inzet op documentatie en online leerplatformen zoals IBM Z Xplore, blijft het tekort van ervaren mensen nog steeds te laag.

De mainframe wereld zal zich dus moeten aanpassen aan de vaardigheden van

de mensen door over te schakelen naar bekendere manieren van werken. Python bijvoorbeeld is een programmeertaal die steeds meer populariteit krijgt sinds zijn creatie in de vroege jaren 90 (Johnson, 2023). Dit is niet enkel bij reeds ervaren programmeurs, maar mensen die net beginnen programmeren kiezen hier steeds vaker voor. Dit is vooral door zijn begin vriendelijkheid, verschillende doeleinden en een actieve community. (Johnson, 2023)

IBM heeft dit opgemerkt en een Python compiler en interpreter ontwikkeld voor z/OS genaamd IBM Open Enterprise SDK for Python. Hierdoor kun je met Python interactie hebben met z/OS om bijvoorbeeld applicaties te ontwikkelen of de resources van het systeem te gaan beheren. Door de Unix omgeving heeft de programmeur ook geen speciale z/OS kennis nodig. (Klaey, 2023)

A.2.2. Bedrijven in het werkveld

Een bank is een goed voorbeeld van een bedrijf die gebruik maakt van IBM hun mainframes. Dit valt te concluderen door het feit dat maar liefst 92 van de top 100 banken wereldwijd een mainframe gebruikt. Dit is ook logisch aangezien er gemiddeld 12.6 miljard financiële transacties per dag uitgevoerd worden. (Wagle, 2017)

Door dit aantal is de nood voor een mainframe toch niet te onderdrukken maar het is niet enkel het aantal transacties dat deze machine kan uitvoeren, het is ook de snelheid, schaalbaarheid en beveiliging van deze systemen dat een grote rol speelt. Het zijn ook niet enkel banken die van deze technologie gebruik maken, maar ook verzekeringsmaatschappijen bijvoorbeeld. De top 10 van deze soort bedrijven maken allemaal gebruik van een IBM mainframe. (Tozzi, 2022)

A.2.3. De Skill gap

Het is nog steeds moeilijk om nieuw talent aan te trekken in de mainframe wereld. Een onderzoek van Deloitte (2020) toont aan dat 79% van de projectleiders moeilijkheden heeft met het zoeken naar mensen met de juiste skillset. Hetzelfde onderzoek toont ook aan dat er in de teams zelf een groot verschil is van kennis en vaardigheden. Hoewel dit systeem gebruikt wordt door 71% van de fortune 500 companies (Tozzi, 2022) , is de Skill gap nog steeds te groot waardoor veel bedrijven vrezen voor een groot tekort aan werknemers om deze z Systems te onderhouden.

Volgens Petra Goude (2023) zijn er verschillende manieren om dit probleem aan te pakken. Zo kunnen bedrijven lessen geven over mainframes en hoe ze dit gebruiken. Ze vertelt ook dat de vaardigheden die nodig zijn beter gecomplimenteerd moeten worden en kunnen leiden naar een belonende en lange termijn carrière.

Hoewel dit zou helpen, vind ze dit niet de kern van het probleem. Het zijn de oude technieken die mensen niet aantrekt. Ze stelt voor om meer te investeren in hedendaagse technologie en dit te installeren op de mainframe. Dit kan gaan over dezelfde test -en deployment technieken, maar ook over hedendaagse programmeertalen zoals Java of Python. Dit zou kunnen door middel van APIs en zou een nieuwe, jongere werkkracht aantrekken. (Goude, 2023)

A.2.4. Toch nog een kleurrijke toekomst

Ondanks de grote nood aan mensen met de juiste vaardigheden, ziet de toekomst er toch nog goed uit. Zo wordt er meer gemoderniseerd met bijvoorbeeld modernere programmeertalen. Er wordt ook voorspeld dat we een introductie van DevOps en self-service benaderingen gaan zien. (Pennaz, 2023)

Momenteel zijn Python en Java beschikbaar als programmeertaal op de mainframe naast PL/1 en COBOL. Sinds deze introductie zijn al bijna 2/3de van gebruikers op de mainframe Java aan het toepassen op een bepaalde manier. (Watts, 2018)

A.3. Methodologie

Deze paper zal als resultaat een volledig stappenplan geven om Pydev op te stellen in IDz en verdere configuratie om de scripts uit te voeren in IDz.

A.3.1. Fase 1: Literatuurstudie

- **Doel:** Relevante informatie verzamelen van IDz, Pydev en de Unix System Services omgeving.
- **Aanpak:**
  - Opzoeken betrouwbare bronnen
  - Gelijke projecten onderzoeken
  - Overzicht van nodige software
- **Tijd:** 5 weken
- **Opbrengst:** Een volledig onderzoek naar vakliteratuur en nodige software die nodig is. Ook een volledige schets naar de omgeving waarin er wordt gewerkt.

A.3.2. Fase 2: Opstellen Pydev

- **Doel:** Opzetten van de Pydev plug-in in IDz
- **Aanpak:**
  - De opgezochte literatuur bestuderen

- Pydev opstellen met de juiste vereisten
- Testen

- **Tijd:** 3 weken
- **Opbrengst:** Pydev voor functies zoals code completion, syntax check, code analysis, ...
  Bij het openen van een Python programma zal dit ook gebeuren via een Python editor

A.3.3. Fase 3: Python interpreter opzetten

- **Doel:** Python Interpreter opstellen om Python programma's uit te voeren in IDz.
- **Aanpak:**

- De opgezochte literatuur bestuderen met expert
- Opzetten van de juiste interpreter
- Testen

- **Tijd:** 2 weken
- **Opbrengst:** Python programma's kunnen uitvoeren in IDz.

A.3.4. Fase 4: Verdere configuratie om de output van de mainframe in de IDz console te krijgen

- **Doel:** Onderzoeken welke functies er beschikbaar zijn in Pydev
- **Aanpak:**

- Pydev documentatie bekijken
- Zelf onderzoeken in IDz

- **Tijd:** 2 weken
- **Opbrengst:** Een overzicht van alle functies die beschikbaar zijn.

A.3.5. Fase 5: Evaluatie voorbereiding

- **Doel:** Eind evaluatie voorbereiden
- **Aanpak:**

- Resultaat bespreken, vergelijken en concluderen
- Op orde stellen van nodige documenten

- **Tijd:** 2 weken
- **Opbrengst:** Een eind evaluatie waarin de opdracht besproken wordt

A.4. Verwacht resultaat, conclusie

Een volledig stappenplan voor de opstelling van Pydev en een Python interpreter die compatibel is met IDz. Ook zal de configuratie getoond worden om de output van Python programma's te zien in IDz. Hierbij zal er bij elke stap uitleg gegeven worden over waarom het op die bepaalde manier gedaan wordt. Ook zal er besproken worden wat er mogelijks mis zou kunnen gaan. Screenshots zullen gebruikt worden voor duidelijk te maken wat er gebeurd.

Door de Pydev plug-in zal de interpreter eenvoudig opgesteld kunnen worden. Hier worden wel wat problemen verwacht omdat Pydev voor standaard Eclipse is ontwikkeld en IDz is hier een afstamming van waardoor er bepaalde onderdelen anders kunnen zijn.

Bibliografie

BasuMallick, C. (2023). What Is a Mainframe? Features, Importance, and Examples. Verkregen maart 28, 2024, van https://www.spiceworks.com/tech/tech-101/articles/what-is-mainframe/

Bostian, J., & Rivera, E. (2023). Securely Leverage Open-Source Software with Python AI Toolkit for IBM z/OS. Verkregen mei 7, 2024, van https://www.redbooks.ibm.com/redpapers/pdfs/redp5709.pdf

Codecadamy. (2022). File System Structure. Verkregen mei 10, 2024, van https://www.codecademy.com/resources/docs/general/file-system-structure

Deloitte. (2020, juni 1). *Mainframe talent drain* (Survey). https://www2.deloitte.com/content/dam/Deloitte/us/Documents/Alliances/us-alliances-deloitte-ibm-mainframe-market-pulse-survey.pdf

Dhawan, A. (2013). What is USS? Verkregen mei 3, 2024, van https://www.zmainframes.com/viewtopic.php?t=49

Eclipse. (2006). What is a wizard? Verkregen mei 7, 2024, van https://wiki.eclipse.org/FAQ_What_is_a_wizard?

George, A. (2021). What Are Plugins, and How Do They Work? Verkregen mei 7, 2024, van https://www.lifewire.com/what-are-plugins-4582189

Goude, P. (2023). 5 ways to bridge the Mainframe skills gap. Verkregen december 13, 2023, van https://www.informationweek.com/it-infrastructure/5-ways-to-bridge-the-mainframe-skills-gap

Hanna, K. (2021). Eclipse (Eclipse Foundation). Verkregen mei 7, 2024, van https://www.techtarget.com/searchapparchitecture/definition/Eclipse-Eclipse-Foundation

HCL Technologies. (2022). HCL Z Data Tools User's Guide and Reference. Verkregen mei 3, 2024, van https://help.hcltechsw.com/zdt/1.1.0/en/base/hfs.html#:~:text=z/OS%C2%AE%20UNIX%E2%84%A2%20provides%20a%20Hierarchical%20File%20System,contain%20files%20or%20other%20subdirectories.

Henry-Stocker, S. (2017). All you need to know about Unix environment variables. Verkregen mei 15, 2024, van https://www.networkworld.com/article/964109/all-you-need-to-know-about-unix-environment-variables.html

IBM. (z.d.-a). Application Programming on z/OS. Verkregen mei 15, 2024, van https://www.ibm.com/docs/en/zosbasics/com.ibm.zos.zappldev/zappldev_book.pdf

IBM. (z.d.-b). Mainframe Concepts. Verkregen mei 3, 2024, van https://www.ibm.com/docs/en/zosbasics/com.ibm.zos.zmainframe/zmainframe_book.pdf

IBM. (z.d.-c). Resilient server solutions with IBM Z. Verkregen mei 2, 2024, van https://www.ibm.com/z/resiliency

IBM. (z.d.-d). z/OS concepts. Verkregen april 26, 2024, van https://www.ibm.com/docs/en/zosbasics/com.ibm.zos.zconcepts/zconcepts_book.pdf

IBM. (2012). z/OS Distributed File Service zSeries File System Implementation z/OS V1R13. Verkregen mei 3, 2024, van https://www.redbooks.ibm.com/redbooks/pdfs/sg246580.pdf

IBM. (2021a). Network security. Verkregen mei 7, 2024, van https://www.ibm.com/docs/pl/zos/2.4.0?topic=intermediate-network-security

IBM. (2021b). IBM Unveils On-Chip Accelerated Artificial Intelligence Processor. Verkregen mei 10, 2024, van https://newsroom.ibm.com/2021-08-23-IBM-Unveils-On-Chip-Accelerated-Artificial-Intelligence-Processor

IBM. (2022). Announcing IBM z16: Real-time AI for Transaction Processing at Scale and Industry's First Quantum-Safe System. Verkregen mei 10, 2024, van https://newsroom.ibm.com/2022-04-05-Announcing-IBM-z16-Real-time-AI-for-Transaction-Processing-at-Scale-and-Industrys-First-Quantum-Safe-System

IBM. (2023). Introduction to SDSF. Verkregen mei 2, 2024, van https://www.ibm.com/docs/en/zos/2.5.0?topic=guide-introduction-sdsf

IBM. (2024a). Unit testing Enterprise COBOL and PL/I applications. Verkregen mei 2, 2024, van https://www.ibm.com/docs/en/developer-for-zos/16.0?topic=eclipse-unit-testing-enterprise-cobol-pli-applications

IBM. (2024b). What's new in IBM Developer for z/OS and Developer for z/OS Enterprise Edition. Verkregen mei 2, 2024, van https://www.ibm.com/docs/en/developer-for-zos/16.0?topic=zos-whats-new-in-developer

Johnson, A. (2023). Python Popularity: The Rise of A Global Programming Language. Verkregen december 13, 2023, van https://flatironschool.com/blog/python-popularity-the-rise-of-a-global-programming-language/#:~:text=Python%20Popularity%20Over%20The%20Years&text=Stack%20Overflow's%20developer%20survey%20results,as%20the%20most%20popular%20language.

Klaey, W. (2023). Python Programming on z/OS. Verkregen december 13, 2023, van https://www.linkedin.com/pulse/python-programming-zos-walter-klaey/?trackingId=eYKhSv70Tc6dnZLPooMDYQ==

Mertic, J. (2020). Getting Jiggy With It: How a Linux Anniversary Led to the Creation of the Open Mainframe Project. Verkregen december 12, 2023, van https://newsroom.ibm.com/How-a-Linux-Anniversary-Led-to-the-Creation-of-the-Open-Mainframe-Project

Morning Consult & IBM. (2022). 2022 IBM GLOBALFINANCIAL FRAUDIMPACT REPORT. Verkregen mei 10, 2024, van https://filecache.mediaroom.com/mr5mr_ibmnewsroom/193031/MC%20+%20IBM%20Financial%20Fraud%20Study%20-%20Global%20Report%20Updated%203.8.22.pdf

Pennaz, M. (2023). The secret to attracting mainframe talent during a skills crisis. Verkregen december 13, 2023, van https://venturebeat.com/data-infrastructure/secret-to-attracting-mainframe-talent-during-skills-crisis/

Precisely. (2020). The Ultimate Guide to Mainframe Machine Data: SMF Data Beyond. Verkregen mei 3, 2024, van https://www.precisely.com/blog/mainframe/ultimate-guide-mainframe-machine-data-smf

Rivera, E. (2023). Python® AI Toolkit for IBM® z/OS®. Verkregen mei 7, 2024, van https://community.ibm.com/community/user/ibmz-and-linuxone/blogs/evan-rivera/2023/02/24/python-ai-toolkit-for-ibm-zos?communityKey=038560b2-e962-4500-b0b5-e3745175a065

Rupp, M. (2022). AN INTRODUCTION TO Z/OS AND THE IBM COMMON CRYPTOGRAPHIC ARCHITECTURE. Verkregen mei 2, 2024, van https://www.cryptomathic.com/news-events/blog/payment-banking-an-introduction-to-z/os-and-the-ibm-common-cryptographic-architecture

Sayer, P. (2022). IBM's z16 mainframe boasts on-chip AI acceleration. Verkregen mei 7, 2024, van https://www.cio.com/article/307884/ibms-z16-mainframe-boasts-on-chip-ai-acceleration.html

Singhal, A. (2023). Parsing Mainframe files (EBCDIC files). Verkregen mei 7, 2024, van https://medium.com/@amar.singhal/data-migration-from-mainframes-parsing-ebcdic-files-83d4900eb0ab

Spohn, J. (2023). IBM Developer for z/OSEnterprise Edition. Verkregen maart 28, 2024, van https://www.ibm.com/downloads/cas/DJ5P7W3N

Tozzi, C. (2022). 10 Mainframe Statistics That May Surprise You. Verkregen december 13, 2023, van https://www.precisely.com/blog/mainframe/9-mainframe-statistics

Turner, D. (2022). PAYMENT BANKING CRYPTOGRAPHY: THE BENEFITS OF Z/OS THE Z PLATFORM. Verkregen mei 2, 2024, van https://www.cryptomathic.com/news-events/blog/payment-banking-cryptography-an-overview-of-the-benefits-of-z/os-and-the-z-platform

Wagle, L. (2017). The modern mainframe. Verkregen december 12, 2023, van https://www.ibm.com/thought-leadership/institute-business-value/en-us/report/modernmainframe

Watts, S. (2018). Java on the Mainframe: z/OS vs Linux. Verkregen december 13, 2023, van https://www.bmc.com/blogs/java-mainframe-zos-linux/