



Python - DLT - IOTA




Name:	Python – DLT – IOTA IoT-Kühlkettenüberwachung mit DLT	
Klasse:		Datum:

Inhalt

1	IoT-Kühlkettenüberwachung mit DLT	3
2	DLT - Distributed Ledger Technologie	4
2.1	Grundlagen	4
2.2	Welches Hauptproblem löst DLT?	4
3	Was ist IOTA?	5
4	Python und DLT	6
4.1	Installation der IOTA-Client-Bibliothek für Python	6
4.2	Iota-Netze	7
4.3	Verbindung mit dem IOTA-Tangle aufnehmen	8
4.4	Eine Nachricht auf den Tangle schreiben	10
4.5	Den Tangle live beobachten	12
4.6	Eine Nachricht im Tangle suchen	12
4.7	Eine Nachricht mit Inhalt auf den Tangle schreiben	13
4.8	Eine Nachricht mit Testdaten lesen	15
4.9	Eine Nachricht mit Temperaturdaten schreiben	16
4.10	Eine Nachricht mit Temperaturdaten lesen	17
5	Projektaufgabe	18
5.1	Projektstufe 1	18
5.1.1	Scenario	18
5.1.2	Technische Spezifikationen	18
5.1.3	Aufgabenstellung	19
6	ECLASS	20
7	MQTT	21
7.1	Eine Nachricht mit Temperaturdaten per MQTT lesen	21
8	DLT-Anwendungsszenarien	22
9	DLT-Detailwissen	23
9.1	Permissionless DLT	23
9.2	Merkmale	24
9.2.1	Dezentralität	24
9.2.2	Unveränderlichkeit	24
9.2.3	Programmiertes Vertrauen	24
9.2.4	Doppelausgabenproblem ist gelöst	24
9.2.5	Transparenz	24

Name:	Python – DLT – IOTA IoT-Kühlkettenüberwachung mit DLT	 BILDUNGS- ZENTRUM TECHNIK UND GESTALTUNG OLDENBURG
Klasse:		

9.2.6	Sofortige Transaktionen	24
-------	-------------------------------	----

Name:	Python – DLT – IOTA IoT-Kühlkettenüberwachung mit DLT	
Klasse:		

Datum:

1 IoT-Kühlkettenüberwachung mit DLT



Ein Bio-Döner möchte die Kühlkette seines Fleisches überwachen.

Programmieren Sie eine

IoT-DLT-Lösung

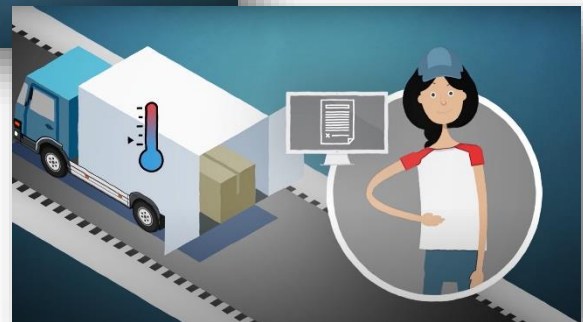
in Python

mit IOTA!




Schauen Sie folgende Film:

<https://youtu.be/L250oRMy8po>



Für die Lösung dieser Aufgabe müssen wir folgende Fragen/Teilaufgaben klären:

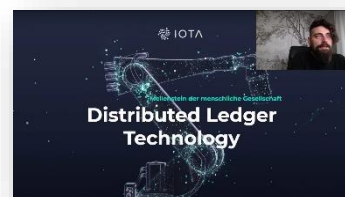
1. Was ist Distributed Ledger Technologie?
2. Was ist IOTA?
3. Wie schreibe ich Daten auf den IOTA-Tangle?
4. Wie lese ich Daten vom IOTA-Tangle?

Name:	Python – DLT – IOTA IoT-Kühlkettenüberwachung mit DLT	
Klasse:		
		Datum:

2 DLT - Distributed Ledger Technologie

Schauen Sie als Einleitung in das Thema folgenden Vortrag:

<https://www.youtube.com/watch?v=pUihldE9os>



2.1 Grundlagen

Der Begriff Distributed-Ledger-Technologie (englisch für Technik verteilter Kassenbücher od. Hauptbücher) ist eine Datenbankarchitektur, die es den Besitzern digitaler Güter ermöglicht, diese von Peer zu Peer zu übertragen und zu dokumentieren. Jede Übertragung in einem DLT wird als Datensatz in einem Distributed-Ledger (Datenbank) gespeichert, diese Datenbank ist in allen Nodes eines Netzwerks gespeichert.

Im Gegensatz zum klassischen Ansatz, bei dem ein Kassenbuch in der Regel von nur einer Instanz verwaltet wird, werden hier dezentral beliebig viele gleichgestellte Kopien des Kassenbuchs von unterschiedlichen Parteien gespeichert. Durch geeignete Maßnahmen wird dafür gesorgt, dass neu hinzuzufügende Transaktionen in allen Kopien des Kassenbuchs übernommen werden und dass es zu einer Übereinkunft (Konsensus) über den jeweils aktuellen Stand des Kassenbuchs kommt.

Es wird auch von dezentral geführten Kontobüchern oder Transaktionsdatenbanken gesprochen. Die Technik gilt als wegweisend für die Verwaltung von Daten im Internet ohne auf eine Eigentümerplattform zurück greifen zu müssen.


Die Distributed-Ledger-Techniken unterscheiden sich durch die Art, wie die vernetzten Computer zu einer Vereinbarung kommen (Konsensus Protokolle), etwa durch „Proof of Work“, durch „Proof of Stake“ und weiteren Verfahren oder Kombinationen.

2.2 Welches Hauptproblem löst DLT?

Problem: Wenn Daten in Eigentümer geführten Datenbanken gespeichert werden, ist es schwierig, diese Daten mit anderen zu teilen, ohne dass sie geändert werden können oder in anderen Datenbanken verloren gehen.

Lösung: DLT schafft eine einzige Wahrheit, auf die alle Teilnehmer vertrauen können. Wenn Daten zu einem verteilten Kassenbuch hinzugefügt werden, kann jeder mit einer Internetverbindung darauf zugreifen, indem er sich mit einem Node im Netzwerk verbindet.

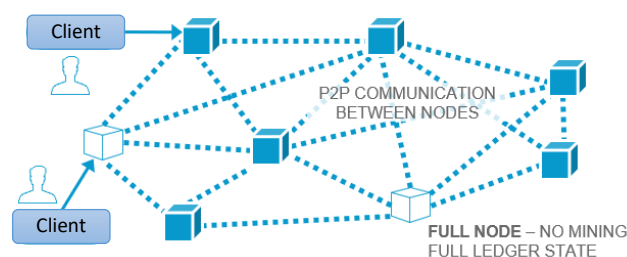
Quelle: <https://iota-einsteiger-guide.de/grundlagen/archiv-distributed-ledger-technologie-dlt/>

Name:	Python – DLT – IOTA IoT-Kühlkettenüberwachung mit DLT	
Klasse:		
		Datum:

3 Was ist IOTA?

Das Ziel der IOTA Foundation ist es, eine Vertrauensschicht (engl. Trust Layer) für das Internet of Everything (IoE) zu erschaffen, die es Geräten im IoE ermöglicht unveränderlich Daten und Werte untereinander gebührenfrei auszutauschen. IOTA strebt in Zusammenarbeit mit der Industrie und der Object-Management-Group, eine Standardisierung ihres Kommunikationsprotokolls an. Mit einer hohen Interoperabilität wird IOTA das “Ledger Of Everything” sein, dessen Infrastruktur auch von externen Anwendungen ohne Erlaubnis genutzt werden kann.

IOTA ermöglicht eine schnelle, manipulationssichere und dezentrale Übertragung von Werten und Daten über viele Nodes, dabei werden Werte- und Daten-Transaktionen grundsätzlich unterschiedlich gehandhabt, während Werte-Transaktionen von Full-Nodes validiert werden müssen, werden Daten-Transaktionen direkt bestätigt und sind notariert.




Jetzt wird sich der ein oder andere Fragen, warum benötige ich für eine reine Daten-Transaktion die IOTA Distributed Ledger Technologie, ich kann die Daten doch einfach verschlüsseln, signieren und via TCP/IP versenden.

Nun, abgesehen davon, dass „Man in the Middle“ Angriffe möglich wären, beweisen signierte Daten nur, dass die Daten von Ihnen kommen. Es erlaubt mir weder zu beweisen, wann Sie sie gesendet haben, noch ob Sie die gleichen Daten an alle gesendet haben. Sie könnten eine bestimmte Information an eine Person und eine andere Information an eine andere Person senden. Signaturen allein werden niemanden vor solchen Dingen schützen.

Mithilfe der „Notarisierung“ kann bewiesen werden, dass ein elektronisches Dokument in einer bestimmten Form zu einem bestimmten Zeitpunkt existiert hat und seit der Erstellung nicht verändert wurde. Bei der Erstellung einer Notarisierung wird ein eindeutiger Hash (Fingerabdruck) eines Dokumentes berechnet und gemeinsam mit einem Zeitstempel im IOTA-Ledger (Tangle) unveränderbar gespeichert. Falls zu einem späteren Zeitpunkt verifiziert werden soll, dass das betreffende Dokument zum behaupteten Zeitpunkt existiert hat und/oder nicht verändert wurde, werden die Daten aus dem Tangle abgerufen und mit den vorliegenden Informationen verglichen.

Unterm Strich bedeutet dies, dass es nicht nur darum geht, dass niemand die Daten während der Übertragung manipuliert, sondern auch darum, dass der Empfänger diese Daten nicht manipuliert. Beispiel: Ein Sensor (mit IDoT Chip) hat einige Werte gemessen/erfasst und versendet diese Daten über den IOTA-Tangle, welcher den Hash dieser Daten speichert. Wenn diese Daten später verkauft werden sollen, kann dieser Hash als Nachweis vorgelegt werden und dem Käufer anhand des Tangle beweisen, dass die Daten vom Sensor im Nachhinein nicht verändert wurden. Die IOTA Technologie (Tangle) fungiert also wie eine Art Fingerabdruck, mit ihm können alle gesendeten Daten verifiziert werden.

Quelle: <https://iota-einsteiger-guide.de/einfuehrung/archiv-was-ist-iota-eine-zusammenfassung/>

Name:	Python – DLT – IOTA IoT-Kühlkettenüberwachung mit DLT	
Klasse:		Datum:

4 Python und DLT

4.1 Installation der IOTA-Client-Bibliothek für Python


Für die Interaktion mit dem IOTA-Tangle über Python muss eine Iota-Client-Bibliothek installiert werden. Hierfür sind vier Schritte notwendig:

1. Eingabeaufforderung (cmd oder powershell) **als Administrator** starten
2. Tool zur Verarbeitung von „wheel“-Dateien installieren mit: `py -m pip install wheel`
3. Wheel-Datei mit Python-Bibliothek herunterladen:
<https://nightly.link/iotaedger/iota.rs/workflows/python-bindings-publish/production>
 Die Datei muss entpackt werden.
 Wechseln Sie in das Verzeichnis, in welches Sie die Datei entpackt haben.
4. Wheel-Datei mit Python-Bibliothek installieren: `py -m pip install <wheel_file>`
 Hinweis: Unbedingt den kompletten Dateinamen inkl. „.whl“ angeben.

Eine ausführliche Installationsanleitung finden Sie hier: https://wiki.iota.org/iota.rs/getting_started/python

Die erfolgreiche Installation kann mit dem Befehl `py -m pip list` überprüft werden.

Package	Version
-----	-----
iota-client-python	0.2.0a3

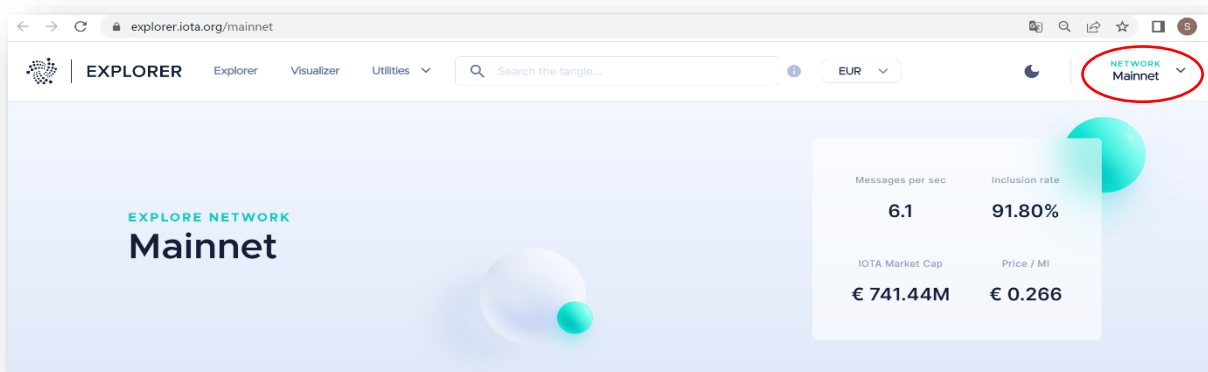
Name:	Python – DLT – IOTA IoT-Kühlkettenüberwachung mit DLT	
Klasse:		
		Datum:

4.2 Iota-Netze

Die Iota-Foundation betreibt aktuell folgende Netzwerke:


- | | |
|--------------------------------|---|
| • Mainnet (Chrysalis) | Aktuelles Hauptnetz mit handelbaren Iota-Tokens (Version Chrysalis) |
| • Chrysalis Devnet (Chrysalis) | Testnetz mit wertlosen Iota-Tokens (Version Chrysalis) |
| • Shimmer (Stardust) | Staging Netz mit handelbaren Shimmer-Tokens (Version Stardust) |
| • Testnet (Stardust) | Testnetz mit wertlosen Shimmer-Tokens (Version Stardust) |
| • Legacy Mainnet (Legacy) | Altes Mainnet vor dem Chrysalis-Update |

Sie können sich die verschiedenen Netze unter <https://explorer.iota.org> anschauen.



Aufgabe

Wie viele Nachrichten werden im Mainnet aktuell pro Sekunde bestätigt? _____

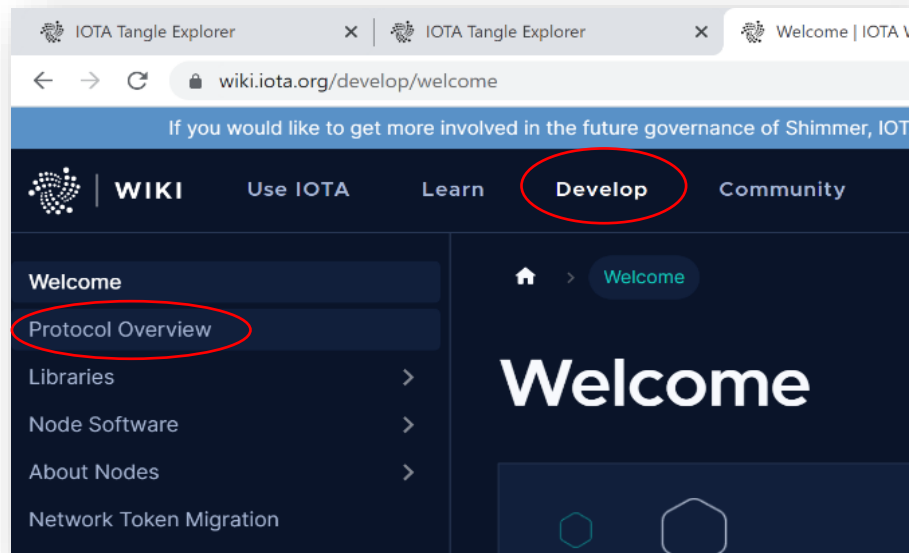
Name:	Python – DLT – IOTA IoT-Kühlkettenüberwachung mit DLT	
Klasse:		

Datum:

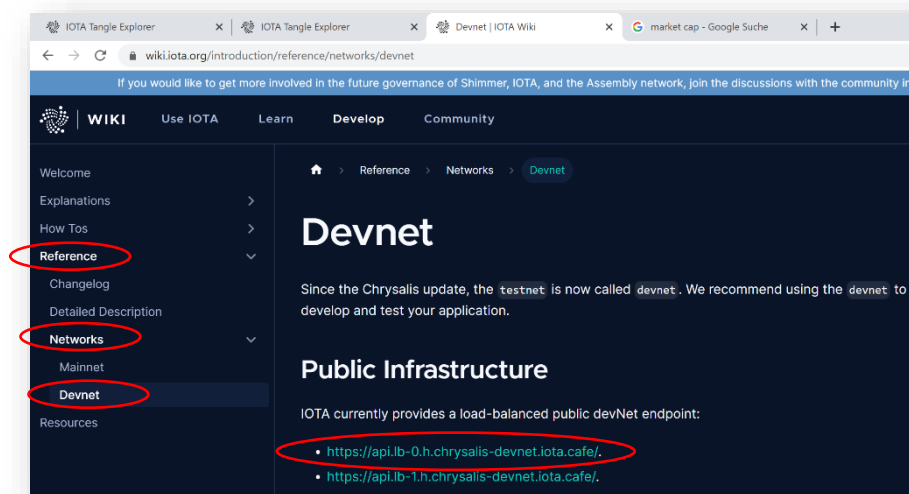
4.3 Verbindung mit dem IOTA-Tangle aufnehmen


Für unsere Testprogramme nutzen wir das „Chrysalis-Devnet“.

Für die Kontaktaufnahme benötigen wir die URL zu einem bestimmten Knoten des Devnets oder die URL zu einem Loadbalancer für das Devnet, der uns dann dynamisch zu einem entsprechenden Knoten weiterleitet.



Die benötigten URLs sind im Iota-Wiki unter „Develop“ => „Protocol Overview“ => dokumentiert:



Name:	Python – DLT – IOTA IoT-Kühlkettenüberwachung mit DLT	
Klasse:		
		Datum:

Verbinden Sie sich zunächst mit einem Knoten und zeigen Sie Informationen zu dem Knoten an.

```
import iota_client

# Knoten aus dem Chrysalis-Devnet
node_url = 'https://api.lb-0.h.chrysalis-devnet.iota.cafe'

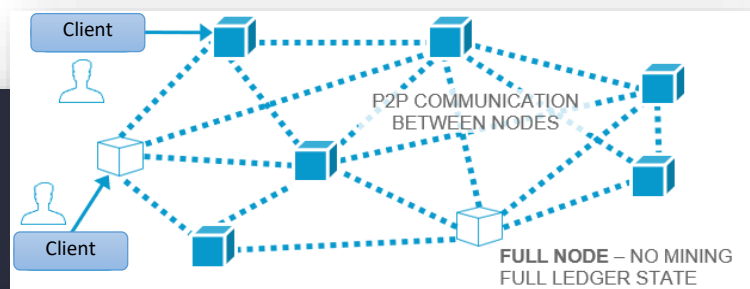
# Client mit einem Knoten verbinden
client = iota_client.Client(nodes_name_password=[[node_url]])


# Knoteninfos ausgeben
print(client.get_info())
```

Eine Anleitung finden Sie hier: https://wiki.iota.org/iota.rs/examples/get_info

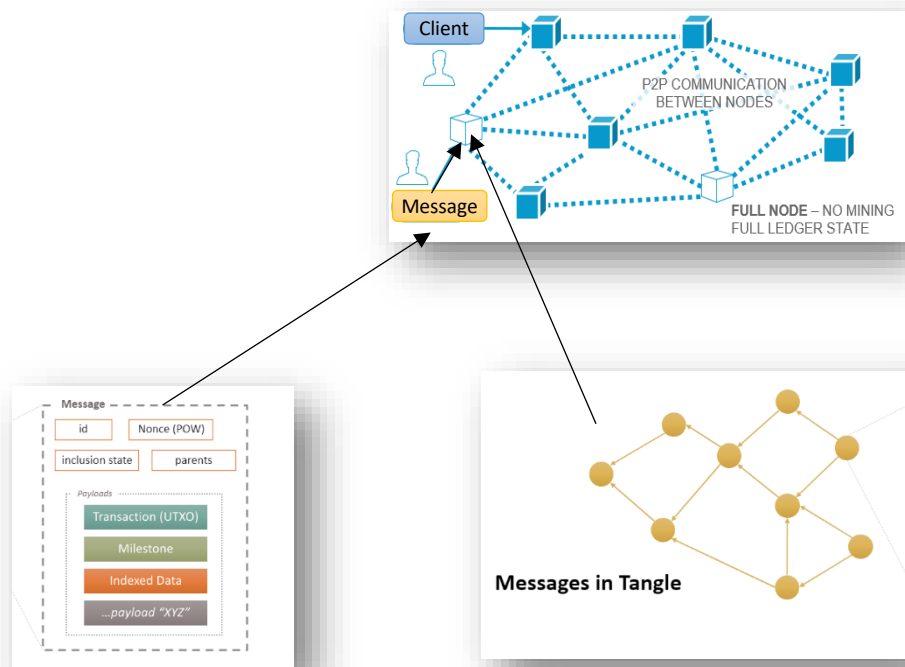
Beispielinfo zu einem Knoten:

```
{
  "nodeinfo": {
    "name": "HORNET",
    "version": "0.6.0-alpha",
    "is_healthy": true,
    "network_id": "migration",
    "bech32_hrp": "atoi",
    "min_pow_score": 100,
    "messages_per_second": 4.2,
    "referenced_messages_per_second": 4.1,
    "referenced_rate": 97.61904761904762,
    "latest_milestone_timestamp": 1618139001,
    "latest_milestone_index": 7092,
    "confirmed_milestone_index": 7092,
    "pruning_index": 0,
    "features": [
      "Pow"
    ]
  },
  "url": "https://api.lb-0.h.chrysalis-devnet.iota.cafe"
}
```



Name:	Python – DLT – IOTA IoT-Kühlkettenüberwachung mit DLT	
Klasse:		
		Datum:

4.4 Eine Nachricht auf den Tangle schreiben



Mit den folgenden Befehlen können Sie eine Message auf den IOTA-Tangle schreiben:

```
import iota_client


# Knoten aus dem Chrysalis-Devnet
node_url = 'https://api.thin-horner-0.h.chrysalis-devnet.iota.cafe'

# Client mit einem Knoten verbinden
client = iota_client.Client(nodes_name_password=[[node_url]])

# Nachricht auf den Tangle schreiben
message = client.message()

# Nachricht ausgeben
print(message)
```

Eine Anleitung finden Sie hier: https://wiki.iota.org/iota.rs/examples/simple_message

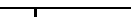
Name:	Python – DLT – IOTA IoT-Kühlkettenüberwachung mit DLT	
Klasse:		Datum:

Als Ergebnis wird die Nachricht im folgenden Format erzeugt und ausgegeben.

```
{
  "message_id": "e2daa4c6b012b615becd6c12189b2c9e701ba0d53b31a15425b21af5105fc086",
  "network_id": 7712883261355838377,
  "parents": [
    "0e2705ce50fec88f896663d4b7d562e74cbcfdd951ac482b1f03cfa5f27396d7",
    "0f5a0b2041766127c3f3bff2dd653b450b72e364765fcc805a40423c59ed01f9",
    "20635b30aee437575d7e6abdf6629eec80543bee30848b0abdda2200fc11a977",
    "da97cd6cfcbb854b8fd3f064c8459c5c9eae80dbd5ef594a3e1a26dcb8fc078c"
  ],
  "payload": "None",
  "nonce": 2305843009213869242
}
```

Die Zeile "payload": "None" sagt aus, dass die Nachricht keinen Inhalt/Wert enthält.

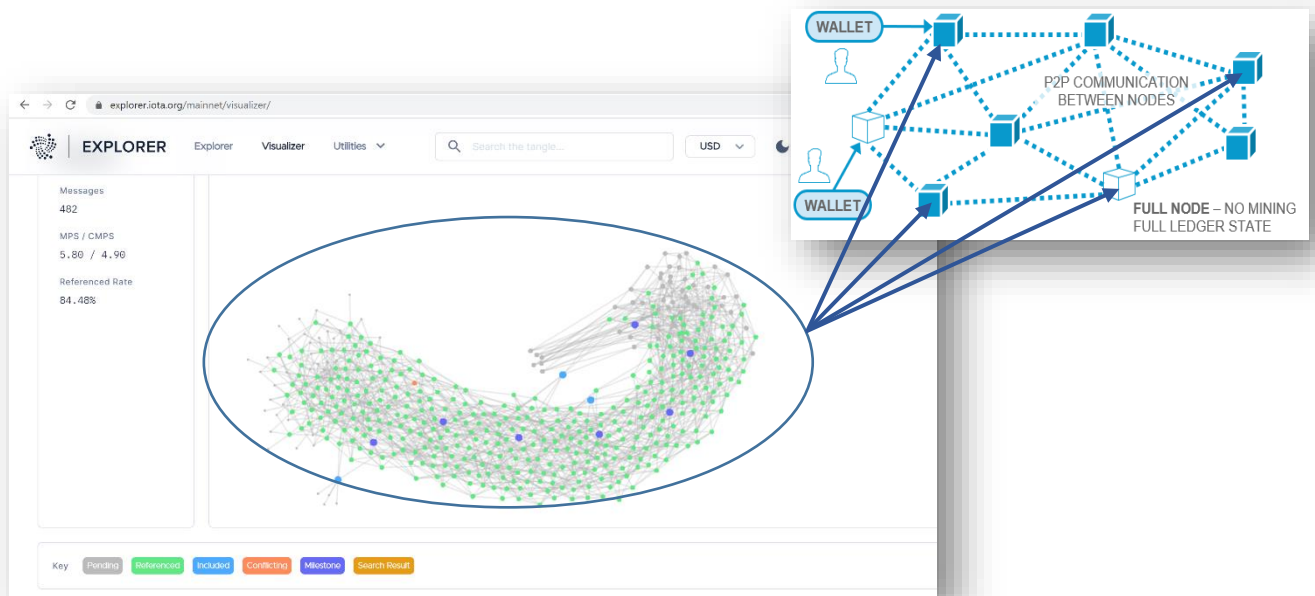
Speichern Sie ihre ausgegebene „Message-ID“ ab, um sie später über Tangle-Explorer im Tangle zu suchen.

Name:	Python – DLT – IOTA IoT-Kühlkettenüberwachung mit DLT	 BILDUNGS ZENTRUM TECHNIK UND GESTALTUNG OLDENBURG
Klasse:		

4.5 Den Tangle live beobachten

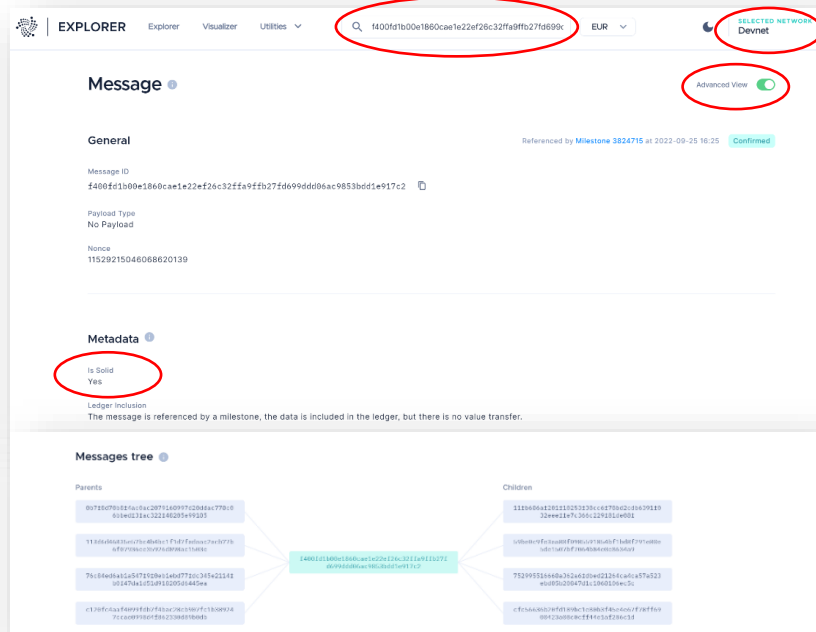
Über folgenden Link: <https://explorer.iota.org/devnet/visualizer/> können Sie die Entwicklung des IOTA-Tangles (bzw. eines Ausschnitts davon) live beobachten.

Der Tangle ist die Datenbank, in der die Nachrichten abgelegt und miteinander verknüpft sind. Der Tangle ist sozusagen das Kassenbuch im DLT. Der Tangle liegt gleichermaßen auf allen Knoten.




4.6 Eine Nachricht im Tangle suchen

Über das Suchfeld und Ihre Message-ID können sie Ihre Nachricht im Tangle anzeigen.



Da die Nachricht den Status „Solid“ besitzt, ist sie durch die DLT bestätigt und kann nicht mehr verändert werden.

Name:	Python – DLT – IOTA IoT-Kühlkettenüberwachung mit DLT	
Klasse:		
		Datum:

4.7 Eine Nachricht mit Inhalt auf den Tangle schreiben

Bitte überlegen Sie sich ein persönliches dreistelliges Kürzel, welches Sie statt „RSE“ benutzen.

Mit den folgenden Befehlen können wir „Testdaten“ unter dem Suchindex „RSE-Test“ auf den Tangle schreiben.

```
import iota_client

# Knoten aus dem Chrysalis-Devnet
node_url = 'https://api.lb-0.h.chrysalis-devnet.iota.cafe'

# Client mit einem Knoten verbinden
client = iota_client.Client(nodes_name_password=[[node_url]])


# Message auf den Tangle schreiben
message = client.message(index = "RSE-Test", data_str = "Testdaten")

# Message ausgeben
print(message)
```

Eine Anleitung finden Sie hier: https://wiki.iota.org/iota.rs/examples/data_message

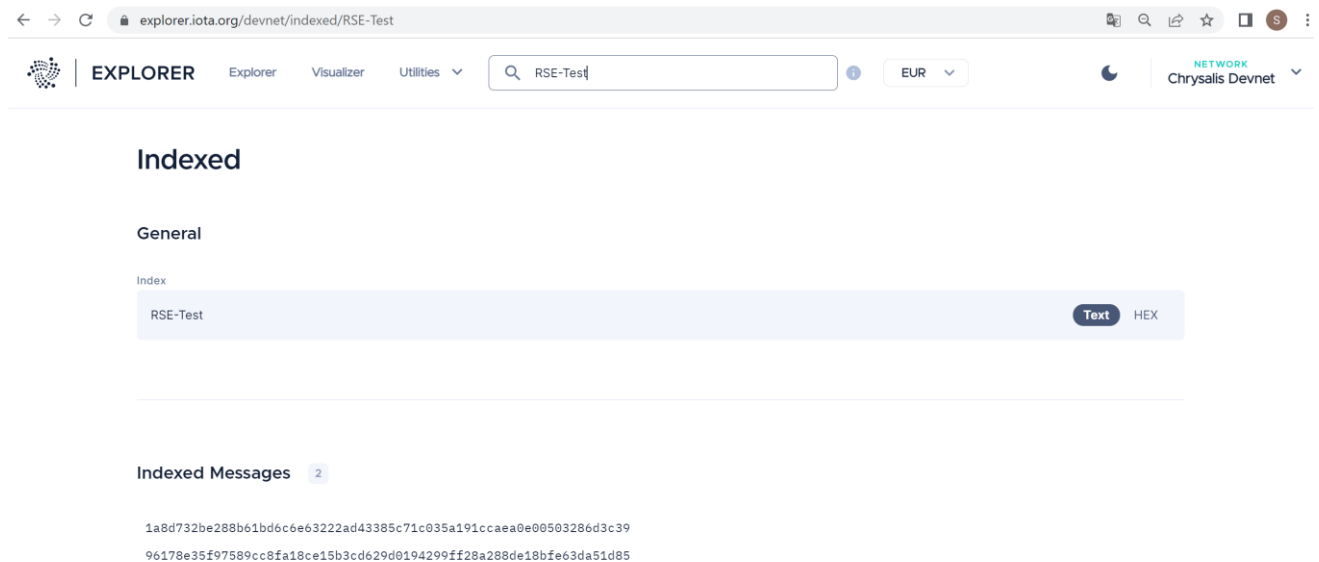
Die gesendete Nachricht sieht folgendermaßen aus:

```
{
  'message_id': 'c71ac44e7fc0eeaa2008aeaf49628a704a706ad9fb9141712e1e513956d8e487',
  'network_id': 6514788332515804015,
  'parents': [
    '8af079c56003713913b00503438fe131b5791b4f2bc858137942002f5c70d892',
    'c92ae1c87b0e980f7f0009313cb176a3e2ed97c7c815c551d02d9667432ee864',
    'caddc645cc8065a725e97f402539e6a551be92ee14f28fa6dfc1ef5ae05b3baf',
    'cdd0cba2d6a3de47a40e412772d4a113fdcdc1fb13ebf113926be284f094a9ef'
  ],
  'payload': {
    'transaction': None,
    'milestone': None,
    'indexation': [{
      'index': '5253452d54657374',
      'data': [84, 101, 115, 116, 100, 97, 116, 101, 110]
    }],
    'receipt': None,
    'treasury_transaction': None
  },
  'nonce': 11529215046068484309
}
```

Name:	Python – DLT – IOTA IoT-Kühlkettenüberwachung mit DLT	
Klasse:		


Weitergehende Informationen gibt es hier: <https://wiki.iota.org/learn/about-iota/data-transfer>

Auch diese Nachricht können wir mit dem Visualizer : <https://explorer.iota.org/devnet/visualizer/> auf dem Tangle wiederfinden. Die Nachricht kann entweder anhand der „Message-ID“ oder des Index „RSE-Test“ gesucht werden. Bei der Indexsuche werden alle Nachrichten zu diesem Index angezeigt.



The screenshot shows the IOTA Explorer interface. The browser address bar displays `explorer.iota.org/devnet/indexed/RSE-Test`. The page title is "EXPLORED". The main content area is titled "Indexed" and shows the "General" tab. Under the "Index" section, the index name "RSE-Test" is displayed. Below this, there is a section for "Indexed Messages" with a count of 2. The messages listed are:

- 1a8d732be288b61bd6c6e63222ad43385c71c035a191ccaea0e00503286d3c39
- 96178e35f97589cc8fa18ce15b3cd629d0194299ff28a288de10bfe63da51d85

Name:	Python – DLT – IOTA IoT-Kühlkettenüberwachung mit DLT	
Klasse:		Datum:

4.8 Eine Nachricht mit Testdaten lesen

```
import iota_client

# Knoten aus dem Chrysalis-Devnet
node_url = 'https://api.lb-0.h.chrysalis-devnet.iota.cafe'


# Client mit einem Knoten verbinden
client = iota_client.Client(nodes_name_password=[[node_url]])

# Messages zu einem Index suchen
mlist = client.find_messages(indexation_keys=['RSE-Test'])

# Schleife über alle gefundenen Messages
for message in mlist:
    # Daten des ersten Eintrages => Liste mit Dezimalwert der Einzelzeichen
    pl = message['payload']['indexation'][0]['data']

    # Dezimalwerte in String mit ASCII-Zeichen umwandeln
    s = ''.join(chr(val) for val in pl)

    print('RSE-Test: ' + s)
    print('-----')
```


Name:	Python – DLT – IOTA IoT-Kühlkettenüberwachung mit DLT	
Klasse:		Datum:

4.9 Eine Nachricht mit Temperaturdaten schreiben

```
import iota_client
from datetime import datetime
import json

# Knoten aus dem Chrysalis-Devnet
node_url = 'https://api.lb-0.h.chrysalis-devnet.iota.cafe'


# Sensordaten als dict anlegen
payload = {
    'sensor-id': 'TS1',
    'timestamp': datetime.now().strftime("%d.%m.%Y %H:%M:%S"),
    'temperature': 7.2
}

# dict als JSON-String ablegen
jspayload = json.dumps(payload)

# Client mit einem Knoten verbinden
client = iota_client.Client(nodes_name_password=[[node_url]])

# Message auf den Tangle schreiben
message = client.message(index="RSE-TS1", data_str = jspayload)

# Message ausgeben
print(message)
```

Name:	Python – DLT – IOTA IoT-Kühlkettenüberwachung mit DLT	
Klasse:		Datum:

4.10 Eine Nachricht mit Temperaturdaten lesen


```
import iota_client
import json

# Knoten aus dem Chrysalis-Devnet
node_url = 'https://api.lb-0.h.chrysalis-devnet.iota.cafe'

# Client mit einem Knoten verbinden
client = iota_client.Client(nodes_name_password=[[node_url]])

# Messages zu einem Index suchen
mlist = client.find_messages(indexation_keys=["RSE-TS1"])

for message in mlist:
    pl = message['payload']['indexation'][0]['data']
    s = ''.join(chr(val) for val in pl)
    d = json.loads(s)
    for key in d:
        print(key + ': ' + str(d[key]))
    print('-----')
```

Name:	Python – DLT – IOTA IoT-Kühlkettenüberwachung mit DLT	
Klasse:		
		Datum:

5 Projektaufgabe

5.1 Projektstufe 1

5.1.1 Scenario


- Der Hersteller „Food Solution Hildesheim“ des Dönerspießes bietet seinen Endkunden eine zertifizierte Kühlkette für alle Produkte an. Diese kann vom Endkunden leicht über einen QR-Code abgefragt werden.
- Jeder Dönerspieß bekommt vom Logistikunternehmen eine eindeutige Identifikationsnummer als QR-Code.
- Jeder Dönerspieß durchläuft während des Transports verschiedene Stationen in der Kühlkette. Ein beispielhafter Transportweg könnte folgendermaßen aussehen:
 - Der **Kühltransporter** (KT) holt die Ware ab und transportiert sie zum
 - lokalen Kühlhaus im **Güterverteilzentrum** (GVZ)
 - Ein **Kühltransporter** (KT) bringt die Ware in ein
 - **Güterverteilzentrum** (GVZ) in der Nähe des Endkunden
 - Ein **Kühltransporter** (KT) bringt die Ware zum Endkunden.
- Der Endkunde kann die Einhaltung der Kühlkette überprüfen, indem er den QR-Code einscannt.

5.1.2 Technische Spezifikationen

- Die Zertifizierung der Kühlkette basiert auf den Informationen, die unveränderlich mit Hilfe von DLT abgespeichert werden.
- Jeder Kühlpunkt schreibt beim Ein- und Auschecken des Produktes folgende Daten auf den Tangle:

```
{'transportid':      '72359278599178561029671',
  'transportstation': 'GVZ-Hildesheim-Kühlhaus2-Zone4',
  'category':        'GVZ',
  'direction':        'in'/'out',
  'timestamp':        '05.09.2022 13:12:25'}
```

- Folgende Kriterien sind für die **Einhaltung der Kühlkette** zu überprüfen:
 - **Stimmigkeit der Kühlketteninformationen**
 - Gibt es für jede Transportstation jeweils einen Eintrag für das Ein-bzw. Auschecken?
 - Sind die Einträge zeitlich sinnvoll geordnet?
 - **Zeiträume ohne Kühlung**
 - Überschreitet die Zeit zwischen dem Auschecken aus einer Transportstation und dem Einchecken in die darauffolgende 10 min?
 - **Transportdauer**
 - Überschreitet die Gesamttransportdauer 48 h?

Name:	Python – DLT – IOTA IoT-Kühlkettenüberwachung mit DLT	
Klasse:		Datum:


5.1.3 Aufgabenstellung

4 x 6er-Gruppe mit Partnerarbeit zu den drei Kriterien

Erstellen Sie ein Programm, dass die benötigten Daten anhand der übergebenen TransportId aus dem IOTA-Tangle ausliest, und die Einhaltung der drei Bedingungen „Stimmigkeit der Kühlketteninformationen“, „Zeiträume ohne Kühlung“ und „Transportdauer“ überprüft. Das Programm soll dem Anwender die Information zurückgeben, ob alle Bedingungen an die Kühlkette erfüllt wurden oder nicht. Falls nicht, soll eine eindeutige Fehlerbeschreibung ausgegeben werden.

Ermitteln Sie, mit Hilfe Ihres Programms, die Ergebnisse für den Index „Food Solution Hildesheim“ und die folgenden TransportId's:

1. 72359278599178561029675
2. 15668407856331648336231
3. 73491878556297128760578
4. 99346757838434834886542
5. 46204863139457546291334
6. 77631003455214677542311
7. 34778534098134729847267
8. 64296734612883933474299
9. 84356113249506843372979
10. 23964376768701928340034
11. 55638471099438572108556
12. 84552276793340958450995
13. 96853785349211053482893
14. 68345254400506854834562
15. 67424886737245693583645
16. 85746762813849598680239
17. 56993454245564893300000
18. 95662334024905944384522
19. 13456783852887496020345
20. 76381745965049879836902

Name:	Python – DLT – IOTA IoT-Kühlkettenüberwachung mit DLT	
Klasse:		
		Datum:

6 ECLASS

Beschreibung von digitalen Zwillingen




D&TS Dataservices & IT-Solutions

Beratung ▾ Dataservice ▾ IT-Solutions ▾ Aktuelles ▾ Referenzen ▾ Unternehmen ▾

ECLASS als Motor der Digitalisierung

Treiben Sie die Digitalisierung in Ihrem Unternehmen voran!
Setzen Sie ECLASS als leistungsfähiges Tool für die Vernetzung und den digitalen Informationsaustausch zwischen den Maschinen und mit Ihren Kunden/Lieferanten ein.

[Jetzt Kontakt aufnehmen](#)

Name:	Python – DLT – IOTA IoT-Kühlkettenüberwachung mit DLT	
Klasse:		

Datum:

7 MQTT

7.1 Eine Nachricht mit Temperaturdaten per MQTT lesen

```
import iota_client
import json
import queue


# The node mqtt url
node_url = 'https://api.lb-0.h.chrysalis-devnet.iota.cafe/'

# The MQTT broker options
broker_options = {
    'automatic_disconnect': True,
    'timeout': 5,
    'use_ws': True,
    'port': 443,
    'max_reconnection_attempts': 5,
}

def on_mqtt_event(event):
    """Put the received event to queue.
    """
    q.put(event)

if __name__ == '__main__':
    client = iota_client.Client(nodes_name_password=[[node_url]], mqtt_broker_options=broker_options)
    client.subscribe_topics(['messages/indexation/RSE-TS1'], on_mqtt_event)

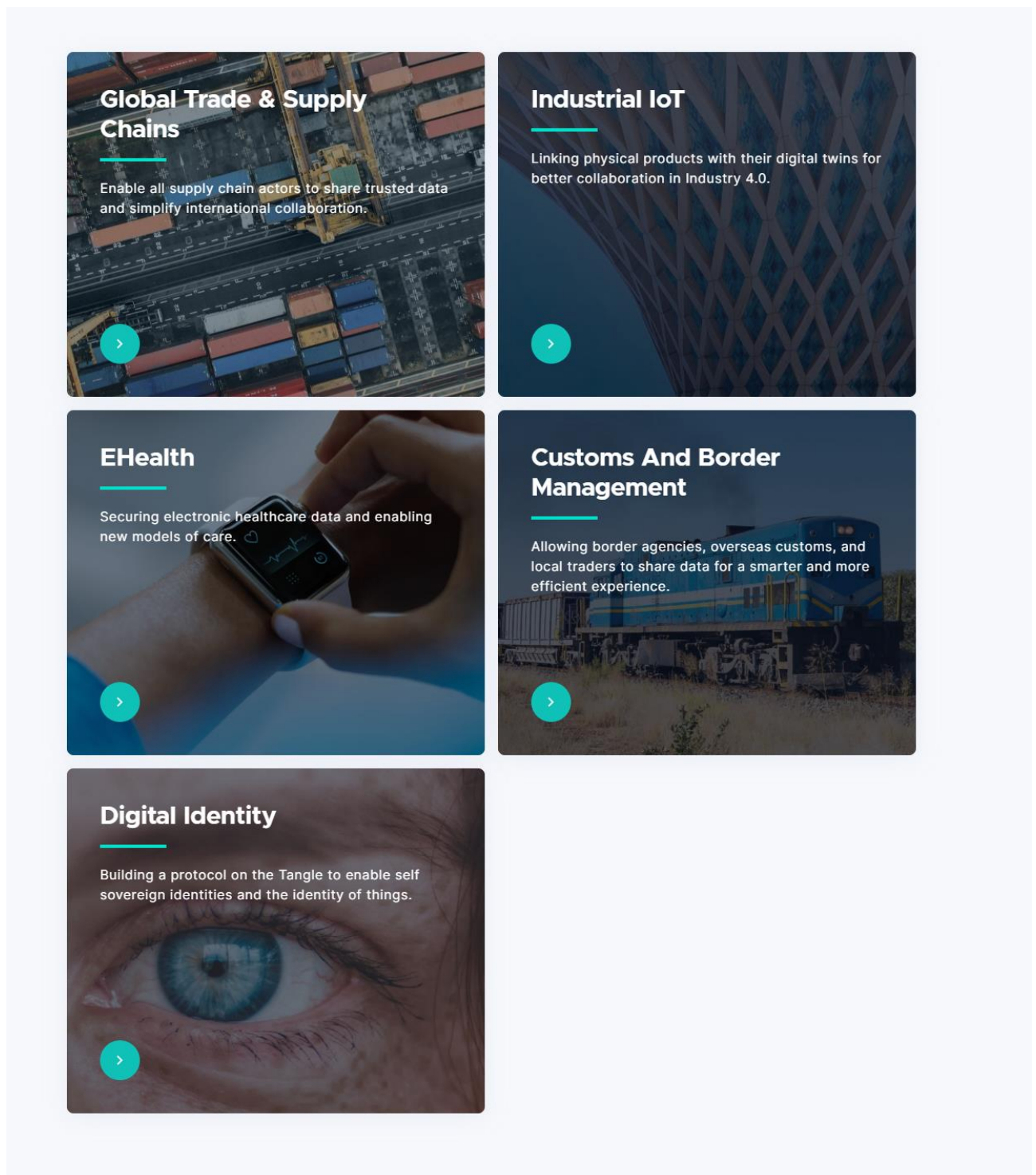
    received_events = 0
    q = queue.Queue()          # The queue to store received events
```


Name:	Python – DLT – IOTA IoT-Kühlkettenüberwachung mit DLT	
Klasse:		
		Datum:

8 DLT-Anwendungsszenarien

Unter folgendem Link, werden erste reale Anwendungsszenarien für den IOTA-Tangle vorgestellt:

<https://www.iota.org/solutions/industries>



Name:	Python – DLT – IOTA IoT-Kühlkettenüberwachung mit DLT	
Klasse:		Datum:


9 DLT-Detailwissen

9.1 Permissionless DLT

Ein DLT kann zwei Arten von Hauptbüchern haben:

Permissionless Ledger (dt. Zulassungsloses Kassenbuch): Ein Kassenbuch, das auf Nodes verteilt ist, die von jedermann ohne Erlaubnis ausgeführt werden können. Der Zweck eines zulassungslosen Kassenbuchs besteht darin, jedem zu ermöglichen, Daten zum Kassenbuch beizutragen, und für jedem, der das Kassenbuch besitzt, identische Kopien aller darin aufgeführten Transaktionen zu geben. Nodes erhalten die Integrität des Kassenbuchs aufrecht, indem ein Konsens über seinen Zustand erzielt wird. Ein zulassungsloses Ledger kann als unabänderlicher globaler Datensatz für Übertragungen verwendet werden.

Permissioned ledger (dt. Zugelassenes Kassenbuch): Ein Kassenbuch, das nur an Nodes verteilt wird, die von einer zentralen Behörde wie einer Bank oder einer Regierung vorgewählt werden.

Name:	Python – DLT – IOTA IoT-Kühlkettenüberwachung mit DLT	
Klasse:		Datum:

9.2 Merkmale

Die wichtigsten Merkmale und Vorteile sind:

9.2.1 Dezentralität

Das Netzwerk wird von keiner zentralen Instanz kontrolliert, alle Netzwerkteilnehmer sind gleichberechtigt, daher kann das Netzwerk nicht von einem einzelnen Akteur abgeschaltet werden. Zudem besteht in einem dezentralen Netzwerk kein Single Point of Failure (dt. einzelner Ausfallpunkt) in dem technischen System, dessen Ausfall den Ausfall des gesamten Systems nach sich zieht, dies erhöht gleichzeitig auch die Sicherheit gegen potentielle Angreifer. Ein dezentrales Netzwerk ist sicherer gegen Manipulationen, es gibt eigene Validierungs- und Autorisierungsmechanismen quer durch das gesamte Netzwerk. Fälschungssichere mathematische Hash-Verfahren machen die Daten vertrauenswürdig. Die Integrität ist sichergestellt, da tausende von Nodes jede Transaktion validieren.

9.2.2 Unveränderlichkeit

Einmal durchgeführte Transaktionen können nicht verändert oder rückgängig gemacht werden. Bei der DLT gibt es nur eine einzige „Quelle der Wahrheit“. Jede unautorisierte Änderung im Netzwerk wird direkt offengelegt, sodass man sich der Richtigkeit der Transaktionen absolut sicher sein kann.

9.2.3 Programmiertes Vertrauen

Vertrauen, welches vormals von Vermittlern (Banken, Händler, etc.) als Dienstleistung angeboten wurde, kann nun technisch zur Verfügung gestellt werden. Seit der Finanzkrise 2008 hat das Vertrauen in die traditionellen Institutionen der Wirtschaft, des Finanzmarktes und sogar des Staates deutlich abgenommen. Mit der DLT können zwei Personen oder Unternehmen, die einander nicht kennen, miteinander ins Geschäft kommen ohne dem anderen vertrauen zu müssen und das auch noch ohne Vermittler. Das spart Zeit (keine Verträge etc.) und Geld (keine Vermittler Provisionen).

9.2.4 Doppelausgabenproblem ist gelöst

Einer der Hauptvorteile der dezentralen Technologien besteht darin, dass das Doppelausgabenproblem gelöst wird. Erklärung: Da digitales Geld nur eine Computerdatei ist, ist es einfach, durch einfaches Kopieren und Einfügen zu fälschen. Ohne DLT verfolgen die Banken das Geld aller Akteure auf ihren Konten, sodass niemand „doppelte Ausgaben“ bzw. das gleiche Geld zweimal ausgeben kann. Die DLT löst dieses Problem anders und effizienter als Banken: Es macht alle Transaktionen und Konten öffentlich, sodass es sofort offensichtlich ist, wenn Geld zweimal verwendet wird.

9.2.5 Transparenz

Jeder Nutzer kann alle Transaktionen nachverfolgen, es ist ein völlig transparentes System. Hinweis: Es gibt auch sogenannte „privacy“ Lösungen.

9.2.6 Sofortige Transaktionen

Transaktionen benötigen viel weniger Zeit als Transaktionen, bei denen eine Art Zwischenhändler (Bank) erforderlich ist.

Quelle: <https://iota-einsteiger-guide.de/grundlagen/archiv-distributed-ledger-technologie-dlt/>