An Auto-Creation Database Persistence in Java

Frank Appiah

January 7, 2021

An Auto-Creation Database Persistence in Java.

F. Appiah ,*Member, IEEE*

Abstract—This study is on automated creation of database system in MySQL database management system from the toolkit of Java persistence. This table creation is a requirement in proper functioning of an intelligent communication of a computer system in the provision of a mobile service for pervasive devices connected by wireless network.

Index Terms—software physics, , table forms, , system, persistence , data communication, wireless communication.

• *F. Appiah is with Kwame Nkrumah University of Science and Technology, Department of Computer Engineering, Kumasi, Ghana. E-mail: appiahnsiahfrank@gmail.com*

1 INTRODUCTION

The implementation of the database[1,2,3,4] is by the Persistence framework [17, 18,19,20] (javax.persistence.*) of Java[9] Toolkit. In this section of system programming, we will look at the persistence functions enabling the database (tables) creation in the AINRS[21] run-time service. The persistence entity classes are as follows:

• Authenticator

• CWApoint (class that models the database containing students information including their CWA's)

• TimeTable (class that models the database that contains student information relating to their timetable)

• Trail (class that models the database that contains student information relating to their trail courses)

• GeneralInfo (class that models the database that contains general information in the school)

• Student (A class that models the student in the database such details as index number,last name, first-name of a student object)

• SmssvrIn

• SmssvrOut

• RegisterCourse (A class that models the student in the database with such details as the number of registered courses and their details.

• Account

• Admin

2 PERSISTENCE CLASS IMPLEMENTATION

2.1 Authenticator Persistence Entity Class Implementation

The persistence entity class is implemented by annotating the Authenticator class as an @Entity and the name of the table is authenticator indicated by annotation @Table. The table column is created by annotating the field of the class by @Column. They are about four (4) named queries in Authenticator entity class used to run already prepared queries:

- Authenticator.findByPhonenumber
- Authenticator.findByPassword
- Authenticator.findById
- Authenticator.findByStudentid

```
@Entity

@Table(name = "authenticator")

@NamedQueries({@NamedQuery(name =
"Authenticator.findByPhonenumber", query
= "SELECT a FROM Authenticator a WHERE
a.phonenumber = :phonenumber"),
@NamedQuery(name =
"Authenticator.findByPassword", query =
"SELECT a FROM Authenticator a WHERE
a.password = :password"),
@NamedQuery(name =
"Authenticator.findById", query = "SELECT
a FROM Authenticator a WHERE a.id =
:id"), @NamedQuery(name =
"Authenticator.findByStudentid", query =
"SELECT a FROM Authenticator a WHERE
a.studentid = :studentid")})

public class Authenticator implements
Serializable {

private static final long
serialVersionUID = 1L;

@Column(name = "phonenumber", nullable =
false)

private String phonenumber;

@Column(name = "password", nullable =
false)

private String password;

@Id

@Column(name = "id", nullable = false)

private Integer id;

@Column(name = "studentid", nullable =
false)

private String studentid;

}
```

2.2 CWAPoint Persistence Entity Class Implementation

The persistence entity class is implemented by annotating the CWAPoint class as an @Entity and the name of the table is cwapoint indicated by annotation @Table. The table columns is created by annotating the field of the class by @Column. They are about four (4) named queries in CWAPoint entity class used to run already prepared queries:

- Cwapoint.findByStudentYear
- Cwapoint.findBySemester
- Cwapoint.findByCwa
- Cwapoint.findById

```
@Entity

@Table(name = "cwapoint")

@NamedQueries({@NamedQuery(name          =
"Cwapoint.findByStudentYear",    query   =
"SELECT   c   FROM   Cwapoint   c   WHERE
c.studentYear       =       :studentYear"),
@NamedQuery(name                         =
"Cwapoint.findBySemester",     query     =
"SELECT   c   FROM   Cwapoint   c   WHERE
c.semester          =          :semester"),
@NamedQuery(name  =  "Cwapoint.findByCwa",
query  =  "SELECT  c  FROM  Cwapoint  c  WHERE
c.cwa    =    :cwa"),    @NamedQuery(name    =
"Cwapoint.findById",   query   =   "SELECT   c
FROM Cwapoint c WHERE c.id = :id")})

public class Cwapoint implements
Serializable {

@Transient

private PropertyChangeSupport
changeSupport = new
PropertyChangeSupport(this);
private static final long
serialVersionUID = 1L;
@Column(name = "studentYear", nullable =
false)
private String studentYear;
@Column(name = "semester", nullable =
false)
private String semester;
@Column(name = "CWA", nullable = false)
private double cwa;
@Id
@Column(name = "ID", nullable = false)
private Integer id;
    @JoinColumn(name = "fk_studentID",
referencedColumnName = "studentID")
@ManyToOne
private Student fkstudentID;
}
```

2.3 Trail Persistence Entity Class Implementation

The persistence entity class is implemented by annotating the Trail class as an @Entity and the name of the table is trail indicated by annotation @Table. The table columns is created by annotating the field of the class by @Column. They are about three(3) named queries in Trail entity class used to run already prepared queries:

- Trail.findByIndexNumber
- Trail.findByStudentID
- Trail.findByCourseTrail

```
@Entity
@Table(name = "trail")
@NamedQueries({@NamedQuery(name =
"Trail.findByIndexNumber", query =
"SELECT t FROM Trail t
WHERE t.indexNumber = :indexNumber"),
@NamedQuery(name =
"Trail.findByStudentID", query = "SELECT
t FROM Trail t WHERE t.studentID =
:studentID"), @NamedQuery(name =
"Trail.findByCourseTrail", query =
"SELECT t FROM Trail t WHERE
t.courseTrail = :courseTrail")})
public class Trail implements
Serializable {
private static final long
serialVersionUID = 1L;
@Id
@Column(name = "indexNumber", nullable =
false)
private Integer indexNumber;
@Column(name = "studentID", nullable =
false)
private int studentID;
@Column(name = "courseTrail", nullable =
false)
private String courseTrail;
}
```

2.4 TimeTable Persistence Entity Class Implementation

The persistence entity class is implemented by annotating the

TimeTable class as an @Entity and the name of the table is timetable indicated by annotation @Table. The table columns is created by annotating the field of the class by @Column. They are about six (6) named queries in TimeTable entity class used to run already prepared queries:

- Timetable.findByCourseID
- Timetable.findByDay
- Timetable.findByInstantTime
- Timetable.findByStudentYear
- Timetable.findByVenue
- Timetable.findByCourse

```
@Entity

@Table(name = "timetable")

@NamedQueries({@NamedQuery(name          =
"Timetable.findByCourseID",    query    =
"SELECT   t   FROM   Timetable   t   WHERE
t.courseID          =          :courseID"),
@NamedQuery(name = "Timetable.findByDay",
query = "SELECT t FROM Timetable t WHERE
t.day   =   :day"),   @NamedQuery(name   =
"Timetable.findByInstantTime",   query   =
"SELECT   t   FROM   Timetable   t   WHERE
t.instantTime = :instantTime"),

@NamedQuery(name                           =
"Timetable.findByStudentYear",   query   =
"SELECT   t   FROM   Timetable   t   WHERE
t.studentYear       =       :studentYear"),
@NamedQuery(name                           =
"Timetable.findByVenue",  query  = "SELECT
t  FROM  Timetable  t  WHERE  t.venue  =
:venue"),        @NamedQuery(name          =
"Timetable.findByCourse", query = "SELECT
t  FROM  Timetable  t  WHERE  t.course  =
:course")})

public class Timetable implements
Serializable {

private static final long
serialVersionUID = 1L;

@Id

@Column(name = "courseID", nullable =
false)

private String courseID;

@Column(name = "day", nullable = false)

private String day;

@Column(name = "instantTime", nullable =
false)

@Temporal(TemporalType.TIME)

private Date instantTime;

@Column(name = "studentYear", nullable =
false)

private int studentYear;

@Column(name = "venue", nullable = false)

private String venue;

@Column(name = "course", nullable =
false)

private String course;

@OneToMany(mappedBy = "courseID")

                      private
Collection<Registercourse>
registercourseCollection;

}
```

2.5 GeneralInfo Persistence Entity Class Implementation

The persistence entity class is implemented by annotating the GeneralInfo class as an @Entity and the name of the table is generalinfo indicated by annotation @Table. The table columns is created by annotating the field of the class by @Column. They are about two (2) named queries in GeneralInfo entity class used to run already prepared queries:

- Generalinfo.findByInfoType
- Generalinfo.findBy | InfoDate

```
@Entity

@Table(name = "generalinfo")

@NamedQueries({@NamedQuery(name          =
"Generalinfo.findByInfoType",    query   =
"SELECT   g   FROM   Generalinfo   g   WHERE
g.infoType          =          :infoType"),
@NamedQuery(name                           =
"Generalinfo.findByInfoDate",    query   =
"SELECT   g   FROM   Generalinfo   g   WHERE
g.infoDate = :infoDate")})

public class Generalinfo implements
Serializable {

private static final long
serialVersionUID = 1L;

@Lob

@Column(name = "infoDetails", nullable =
false)

private String infoDetails;

@Id

@Column(name = "infoType", nullable =
false)

private String infoType;
@Column(name = "infoDate")
@Temporal(TemporalType.TIMESTAMP)
private Date infoDate;
}
```

2.6 Student Persistence Entity Class Implementation

The persistence entity class is implemented by annotating the Student class as an @Entity and the name of the table is student indicated by annotation @Table. The table columns is created by annotating the field of the class by @Column. They are about seven (7) named queries in student entity class used to run already prepared queries:

- Student.findByStudentID
- Student.findByFirstName
- Student.findByCourse
- Student.findByLastName
- Student.findByStudentYear
- Student.findByIndexNumber
- Student.findByMiddleName

```
@Entity
@Table(name = "student")
@NamedQueries({@NamedQuery(name = "Student.findByStudentID", query = "SELECT s FROM Student s WHERE s.studentID = :studentID"), @NamedQuery(name = "Student.findByFirstName", query = "SELECT s FROM Student s WHERE s.firstName = :firstName"), @NamedQuery(name = "Student.findByCourse", query = "SELECT s FROM Student s WHERE s.course = :course"), @NamedQuery(name = "Student.findByLastName", query = "SELECT s FROM Student s WHERE s.lastName = :lastName"), @NamedQuery(name = "Student.findByStudentYear", query = "SELECT s FROM Student s WHERE s.studentYear = :studentYear"), @NamedQuery(name = "Student.findByIndexNumber", query = "SELECT s FROM Student s WHERE s.indexNumber = :indexNumber"), @NamedQuery(name = "Student.findByMiddleName", query = "SELECT s FROM Student s WHERE s.middleName = :middleName")})
public class Student implements Serializable {
private static final long serialVersionUID = 1L;
@Id
@Column(name = "studentID", nullable = false)
private Integer studentID;
@Column(name = "firstName", nullable = false)
private String firstName;
@Column(name = "course", nullable = false)
private String course;
@Column(name = "lastName", nullable = false)
private String lastName;
@Column(name = "studentYear", nullable = false)
private String studentYear;
@Column(name = "indexNumber", nullable = false)
private int indexNumber;
@Column(name = "middleName")
private String middleName;
@OneToMany(mappedBy = "fkstudentID")
private Collection<Cwapoint> cwapointCollection;
        @OneToMany(cascade = CascadeType.ALL, mappedBy = "studentid")
private Collection<Account> accountCollection;
@OneToMany(mappedBy = "studentID")
                    private Collection<Registercourse> registercourseCollection;
}
```

2.7 SmssvrIn Persistence Entity Class Implementation

The persistence entity class is implemented by annotating the SmssvrIn class as an @Entity and the name of the table is smssvr_in indicated by annotation @Table. The table columns is created by annotating the field of the class by @Column. They are about nine (9) named queries in SmssvrIn entity class used to run already prepared queries:

- SmssvrIn.findById
- SmssvrIn.findByProcess
- SmssvrIn.findByOriginator
- SmssvrIn.findByType
- SmssvrIn.findByEncoding
- SmssvrIn.findByMessageDate
- SmssvrIn.findByReceivedDate
- SmssvrIn.findByText

```
@Entity
@Table(name = "smssvr_in")
@NamedQueries({@NamedQuery(name =
"SmssvrIn.findById", query = "SELECT s
FROM SmssvrIn s WHERE s.id = :id"),
@NamedQuery(name =
"SmssvrIn.findByProcess", query = "SELECT
s FROM SmssvrIn s WHERE s.process =
:process"), @NamedQuery(name =
"SmssvrIn.findByOriginator", query =
"SELECT s FROM SmssvrIn s WHERE
s.originator = :originator"),
@NamedQuery(name = "SmssvrIn.findByType",
query = "SELECT s FROM SmssvrIn s WHERE
s.type = :type"), @NamedQuery(name =
"SmssvrIn.findByEncoding", query =
"SELECT s FROM SmssvrIn s WHERE
s.encoding = :encoding"),
@NamedQuery(name =
"SmssvrIn.findByMessageDate", query =
"SELECT s FROM SmssvrIn s WHERE
s.messageDate = :messageDate"),
@NamedQuery(name =
"SmssvrIn.findByReceiveDate", query =
"SELECT s FROM SmssvrIn s WHERE
s.receiveDate = :receiveDate"),
@NamedQuery(name = "SmssvrIn.findByText",
query = "SELECT s FROM SmssvrIn s WHERE
s.text = :text"), @NamedQuery(name =
"SmssvrIn.findByGatewayId", query =
"SELECT s FROM SmssvrIn s WHERE
s.gatewayId = :gatewayId")})
public class SmssvrIn implements
Serializable {
@Transien
private PropertyChangeSupport
changeSupport = new
PropertyChangeSupport(this);
private static final long
serialVersionUID = 1L;
@Id
@Column(name = "id", nullable = false)
private Long id;
@Column(name = "process")
private Integer process;
@Column(name = "originator")
private String originator;
@Column(name = "type")
private Character type;
@Column(name = "encoding")
private Character encoding;
@Column(name = "message_date")
@Temporal(TemporalType.TIMESTAMP)
private Date messageDate;
@Column(name = "receive_date")
@Temporal(TemporalType.TIMESTAMP)
private Date receiveDate;
@Column(name = "text")
private String text;
@Column(name = "gateway_id")
private String gatewayId;
}
```

2.8 SmssvrOut Persistence Entity Class Implementation

The persistence entity class is implemented by annotating the SmssvrOut class as an @Entity and the name of the table is smss-vr_out indicated by annotation @Table. The table columns is created by annotating the field of the class by @Column. They are about seven (7) named queries in SmssvrOut entity class used to run already prepared queries:

• SmssvrOut.findById

• SmssvrOut.findByRecipient

• SmssvrOut.findByText

• SmssvrOut.findByCreateDate

• SmssvrOut.findByOriginator

• SmssvrOut.findByEncoding

• SmssvrOut.findByStatusReport

```
@Entity
@Table(name = "smssvr_out")
@NamedQueries({@NamedQuery(name =
"SmssvrOut.findById", query = "SELECT s
FROM SmssvrOut s WHERE s.id = :id"),
@NamedQuery(name =
"SmssvrOut.findByRecipient", query =
"SELECT s FROM SmssvrOut s WHERE
s.recipient = :recipient"),
@NamedQuery(name =
"SmssvrOut.findByText", query = "SELECT s
FROM SmssvrOut s WHERE s.text = :text"),
@NamedQuery(name =
"SmssvrOut.findByCreateDate", query =
"SELECT s FROM SmssvrOut s WHERE
s.createDate = :createDate"),
@NamedQuery(name =
"SmssvrOut.findByOriginator", query =
"SELECT s FROM SmssvrOut s WHERE
s.originator = :originator"),
@NamedQuery(name =
"SmssvrOut.findByEncoding", query =
"SELECT s FROM SmssvrOut s WHERE
s.encoding = :encoding"),
@NamedQuery(name =
"SmssvrOut.findByStatusReport", query =
"SELECT s FROM SmssvrOut s WHERE
s.statusReport = :statusReport"),
@NamedQuery(name =
"SmssvrOut.findByFlashSms", query =
"SELECT s FROM SmssvrOut s WHERE
s.flashSms = :flashSms"),
@NamedQuery(name =
"SmssvrOut.findBySrcPort", query =
"SELECT s FROM SmssvrOut s WHERE
s.srcPort = :srcPort"), @NamedQuery(name
= "SmssvrOut.findByDstPort", query =
"SELECT s FROM SmssvrOut s WHERE
s.dstPort = :dstPort"), @NamedQuery(name
= "SmssvrOut.findBySentDate", query =
"SELECT s FROM SmssvrOut s WHERE
s.sentDate = :sentDate"),
@NamedQuery(name =
"SmssvrOut.findByRefNo", query = "SELECT
s FROM SmssvrOut s WHERE s.refNo =
:refNo"), @NamedQuery(name =
"SmssvrOut.findByPriority", query =
"SELECT s FROM SmssvrOut s WHERE
s.priority = :priority"),
@NamedQuery(name =
"SmssvrOut.findByErrors", query = "SELECT
s FROM SmssvrOut s WHERE s.errors =
:errors"), @NamedQuery(name =
"SmssvrOut.findByGatewayId", query =
"SELECT s FROM SmssvrOut s WHERE
s.gatewayId = :gatewayId"),
@NamedQuery(name =
"SmssvrOut.findByStatus", query = "SELECT
s FROM SmssvrOut s WHERE s.status =
:status")})
public class SmssvrOut implements
Serializable {
private static final long
serialVersionUID = 1L;
@Id
@Column(name = "id", nullable = false)
private Long id;
@Column(name = "recipient")
private String recipient;
@Column(name = "text")
private String text;
@Column(name = "create_date")
@Temporal(TemporalType.TIMESTAMP)
private Date createDate;
@Column(name = "originator")
private String originator;
@Column(name = "encoding")
private Character encoding;
@Column(name = "status_report")
private Integer statusReport;
@Column(name = "flash_sms")
private Integer flashSms;
@Column(name = "src_port")
private Integer srcPort;
@Column(name = "dst_port")
private Integer dstPort;
@Column(name = "sent_date")
@Temporal(TemporalType.TIMESTAMP)
private Date sentDate;
@Column(name = "ref_no")
private String refNo;
@Column(name = "priority")
private String priority;
@Column(name = "errors", nullable =
false)
private int errors;
@Column(name = "gateway_id", nullable =
false)
private String gatewayId;
@Column(name = "status")
private Character status;
}
```

2.9 RegisterCourse Persistence Entity Class Implementation

The persistence entity class is implemented by annotating the RegisterCourse class as an @Entity and the name of the table is registercourse indicated by annotation @Table. The table columns is created by annotating the field of the class by @Column. They are about one (1) named queries in RegisterCourse entity class used to run already prepared queries:

• Registercourse.findById

```
@Entity
@Table(name = "registercourse")
@NamedQueries({@NamedQuery(name =
"Registercourse.findById", query =
"SELECT r FROM Registercourse r WHERE
r.id = :id")})
public class Registercourse implements
Serializable {
private static final long
serialVersionUID = 1L;
@Id
@Column(name = "ID", nullable = false)
private Integer id;
@JoinColumn(name = "courseID",
referencedColumnName = "courseID")
@ManyToOne
private Timetable courseID;
@JoinColumn(name = "studentID",
referencedColumnName = "studentID")
@ManyToOne
private Student studentID;
}
```

2.10 Account Persistence Entity Class Implementation

The persistence entity class is implemented by annotating the Account class as an @Entity and the name of the table is account indicated by annotation @Table. The table columns is created by annotating the field of the class by @Column. They are about six (6) named queries in Account entity class used to run already prepared queries:

• Account.findByAmtpaid
• Account.findByBalance
• Account.findByDraftcode
• Account.findByRegcode
• Account.findById
• Account.findByRegistered

```
@Entity

@Table(name = "account")

@NamedQueries({@NamedQuery(name =
"Account.findByAmtpaid", query = "SELECT
a FROM Account a WHERE a.amtpaid =
:amtpaid"), @NamedQuery(name =
"Account.findByBalance", query = "SELECT
a FROM Account a WHERE a.balance =
:balance"), @NamedQuery(name =
"Account.findByDraftcode", query =
"SELECT a FROM Account a WHERE
a.draftcode = :draftcode"),
@NamedQuery(name =
"Account.findByRegcode", query = "SELECT
a FROM Account a WHERE a.regcode =
:regcode"), @NamedQuery(name =
"Account.findByRegistered", query =
"SELECT a FROM Account a WHERE
a.registered = :registered"),
@NamedQuery(name = "Account.findById",
query = "SELECT a FROM Account a WHERE
a.id = :id")})

public class Account implements
Serializable {

private static final long
serialVersionUID = 1L;

@Column(name = "amtpaid", nullable =
false)

private int amtpaid;

@Column(name = "balance", nullable =
false)

private int balance;

@Column(name = "draftcode", nullable =
false)

private String draftcode;

@Column(name = "regcode", nullable =
false)

private String regcode;

@Column(name = "registered")

private Character registered;

@Id

@Column(name = "id", nullable = false)

private Integer id;

@JoinColumn(name = "studentid",
referencedColumnName = "studentID")

@ManyToOne

private Student studentid;

}
```

2.11 Admin Persistence Entity Class Implementation

The persistence entity class is implemented by annotating the Admin class as an @Entity and the name of the table is admin indicated by annotation @Table. The table columns is created by annotating the field of the class by @Column. They are about two (2) named queries in Admin entity class used to run already prepared queries:

- Admin.findByUsername
- Admin.findByPassword

```
@Entity

@Table(name = "admin")

@NamedQueries({@NamedQuery(name =
"Admin.findByUsername", query = "SELECT a
FROM Admin a WHERE a.username =
:username"), @NamedQuery(name =
"Admin.findByPassword", query = "SELECT a
FROM Admin a WHERE a.password =
:password")})

public class Admin implements
Serializable {

private static final long
serialVersionUID = 1L;

@Id

@Column(name = "username", nullable =
false)

private String username;

@Column(name = "password", nullable =
false)

private String password;

}
```

3 Conclusion

This work is a showpiece of the author's study in PhD thesis chapter. This shows how to create database management[8] application in Java with its persistence toolkit. This considered about 11 entity classes in total in developing a mobile-2-computer (M2C) system [5,6,7]. This system[10] responds to students in accessing school activities on their mobile phones over wireless network.

References

[1] Silberschatz, A., Korth, H. F., & Sudarshan, S. (1997). Database system concepts (Vol. 5). New York: McGraw-Hill.
[2] Lamb, C., Landis, G., Orenstein, J., & Weinreb, D. (1991). The ObjectStore database system. Communications of the ACM, 34(10), 50-63.
[3] Garcia-Molina, H., Ullman, J. D., & Widom, J. (2000). Database system implementation (Vol. 672). Upper Saddle River, NJ:: Prentice Hall.
[4] Atkinson, M., Dewitt, D., Maier, D., Bancilhon, F., Dittrich, K., & Zdonik, S. (1990). The object-oriented database system manifesto. In Deductive and object-oriented databases (pp. 223-240). North-Holland.
[5] Labrou, Y., & Agre, J. R. (2009). U.S. Patent No. 7,606,560. Washington, DC: U.S. Patent and Trademark Office.
[6] Song, C. J., Kim, S. P., Seo, M. D., Lee, K. G., & Jin, Y. J. (2012). U.S. Patent Application No. 13/196,357.
[7] Tadayon, S., & Halavi, M. (2012). U.S. Patent No. 8,315,617. Washington, DC: U.S. Patent and Trademark Office.
[8] Starkey, J. A. (2012). U.S. Patent No. 8,224,860. Washington, DC: U.S. Patent and Trademark Office.
[9] Arnold, K., Gosling, J., Holmes, D., & Holmes, D. (2000). The Java programming language (Vol. 2). Reading: Addison-wesley.
[10] Rogers, E. M. (1986). Communication technology. Simon and Schuster.
[11] Lee, E. A., & Messerschmitt, D. G. (2012). Digital communication. Springer Science & Business Media.
[12] Tuohino, M., Poikselka, M., Mayer, G., & Westman, I. (2005). U.S. Patent Application No. 10/932,253.
[13] Gabor, D. (1950). CIII. Communication theory and physics. The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science, 41(322), 1161-1187.
[14] Littlejohn, S. W., & Foss, K. A. (2010). Theories of human communication. Waveland press.
[15] Tomasello, M. (2010). Origins of human communication. MIT press.
[16] Shannon, C. E. (1949). Communication in the presence of noise. Proceedings of the IRE, 37(1), 10-21.
[17] Böck, H. (2012). Java persistence api. In The Definitive Guide to NetBeans™ Platform 7 (pp. 315-320). Apress.
[18] Yang, D. (2010). Java persistence with JPA. Outskirts Press.
[19] Goncalves, A. (2013). Java persistence API. In Beginning Java EE 7 (pp. 103-124). Apress, Berkeley, CA.
[20] Müller, B., & Wehr, H. (2012). Java Persistence API 2: Hibernate, EclipseLink, OpenJPA und Erweiterungen. Carl Hanser Verlag GmbH Co KG.
[21] Appiah, FK (2017). Automatic Information Retrieval System: Intelligent Computer Communication.

Dr. Frank Appiah. He is a holder of Bsc(Hon) from Kwame Nkrumah University of Science and Technology in 2018, Msc in Advanced Software Engineering from King's college London in 2010 and PhD in computer science and engineering from both KCL (2012/2014) and KNUST (2014) respectively. Frank Appiah has professional certificates in Management and engineering since 2011. He developed StreamEPS - Stream Event Processing System in 2011 which is hosted at Github. This work is a PhD thesis at KNUST, Department of Computer Engineering, Kumasi, Ghana. This work is awarded course leadership programme by Kwame Nkrumah University of Science and Technology.