Akademia Górniczo-Hutnicza

Wydział Elektrotechniki, Automatyki, Informatyki i Inżynierii Biomedycznej
Kierunek Informatyka, III rok, 2018/2019

Analiza i modelowanie oprogramowania

Sprawozdanie z projektu

„Gdzie jest moje dziecko?"

Agnieszka Zadworny, Maciej Bielech

Kraków, 23 listopada 2018

1 General description of system

1.1 System's goals

Our application's aim is to provide possibility to control current location of children. Parent can create rules, containing information about area in which child should be in specific period of time. If child breaks the rule, parent will get push notification on smartphone. In case when child turn off the application or GPS, parent will be notified about that incident, and parent may check last registered location of child's phone.

1.2 Stakeholders and their aims

Stakeholder	**Aim**	**Priority**
Parent	checking current location of child	high
Parent	signing in with Google account	moderate
Parent	signing in with Facebook, Twitter, Instagram or another social networking account	low
Parent	checking the history of child's location	low
Parent	have mobile application for Android device	high
Parent	have mobile application for iOS device	moderate
Parent	pay for a week or month	high
Parent	pay in advance for whole year with some discount	moderate
Parent	creating custom shapes of areas	moderate
Parent	creating rules	high
Parent	receiving notifications about broken rules	high
Child	sending current location	high
Child	minimize application without closing connection	high
Child	have mobile application for Android device	high
Child	have mobile application for iOS device	moderate
Team	usage of modern technologies like ReactJS, Redux, NodeJS, MongoDB etc.	moderate
Team	usage of Google Maps API, Google Geofencing API	high
Team	have two servers for web application and database	high

Tablica 1: Stakeholders and their aims

1.3 Borders of system

There are following actors:

- Parent,
- Child,
- Google OAuth system,
- Google Maps system,
- Time.

1.4 List of capabilities

1.4.1 Capabilities of web application:

- Creation of parent's account,
- Signing parent with email,
- Signing parent with googleAuth,
- Signing parent out,

- CRUD areas,
- CRUD children,
- CRUD rules,
- Switching on/off rule's activity,
- Choosing payment's plan,
- Adding payments credentials.

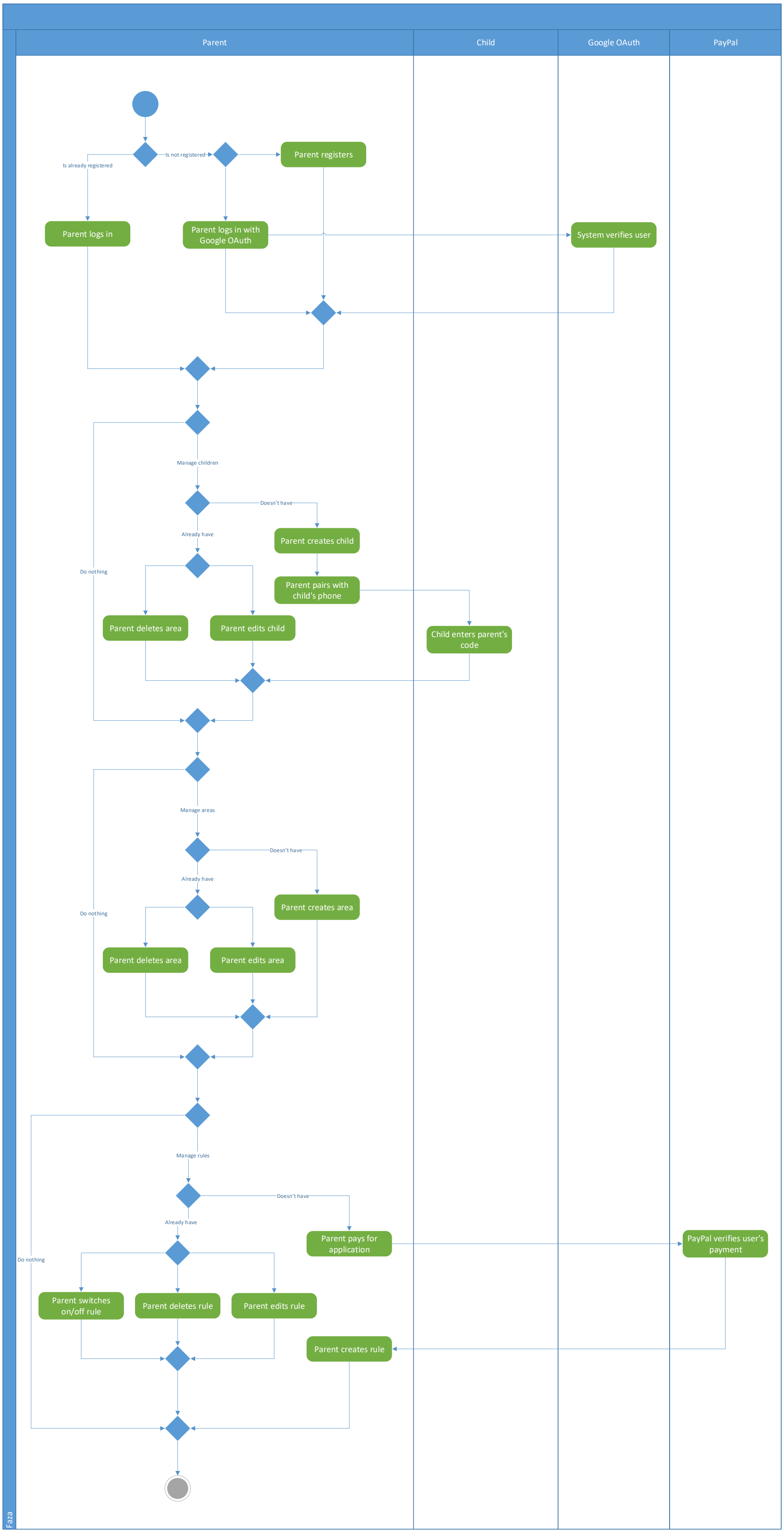
1.4.2 Capabilities of child mobile application:

- Finding child's phone location,
- Sending child's phone location to server,
- Working in the background as service.

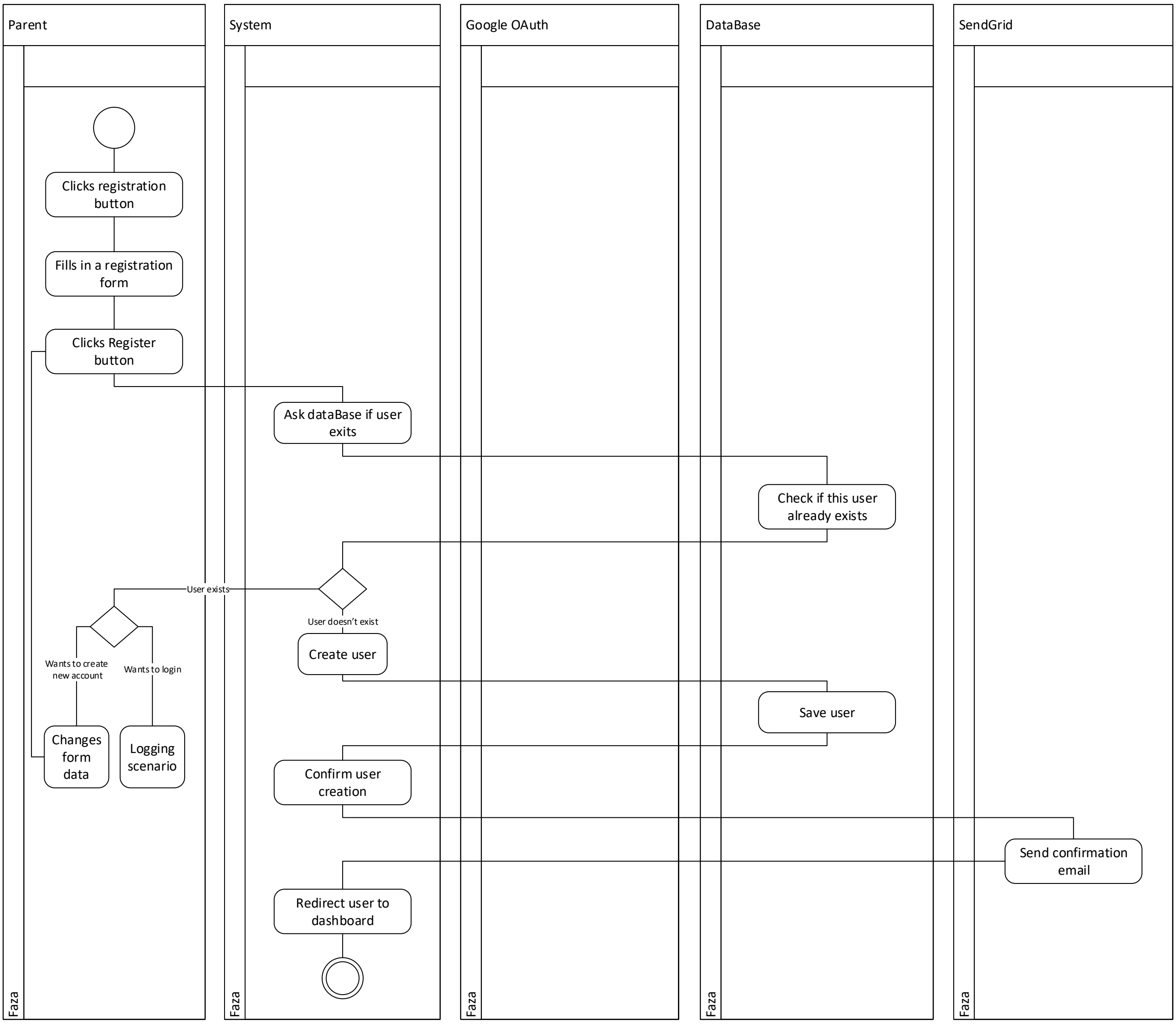
1.4.3 Capabilities of parents mobile application:

- Connecting with server to allow server to send push notifications,
- Displaying last children locations,
- Receiving push notifications.

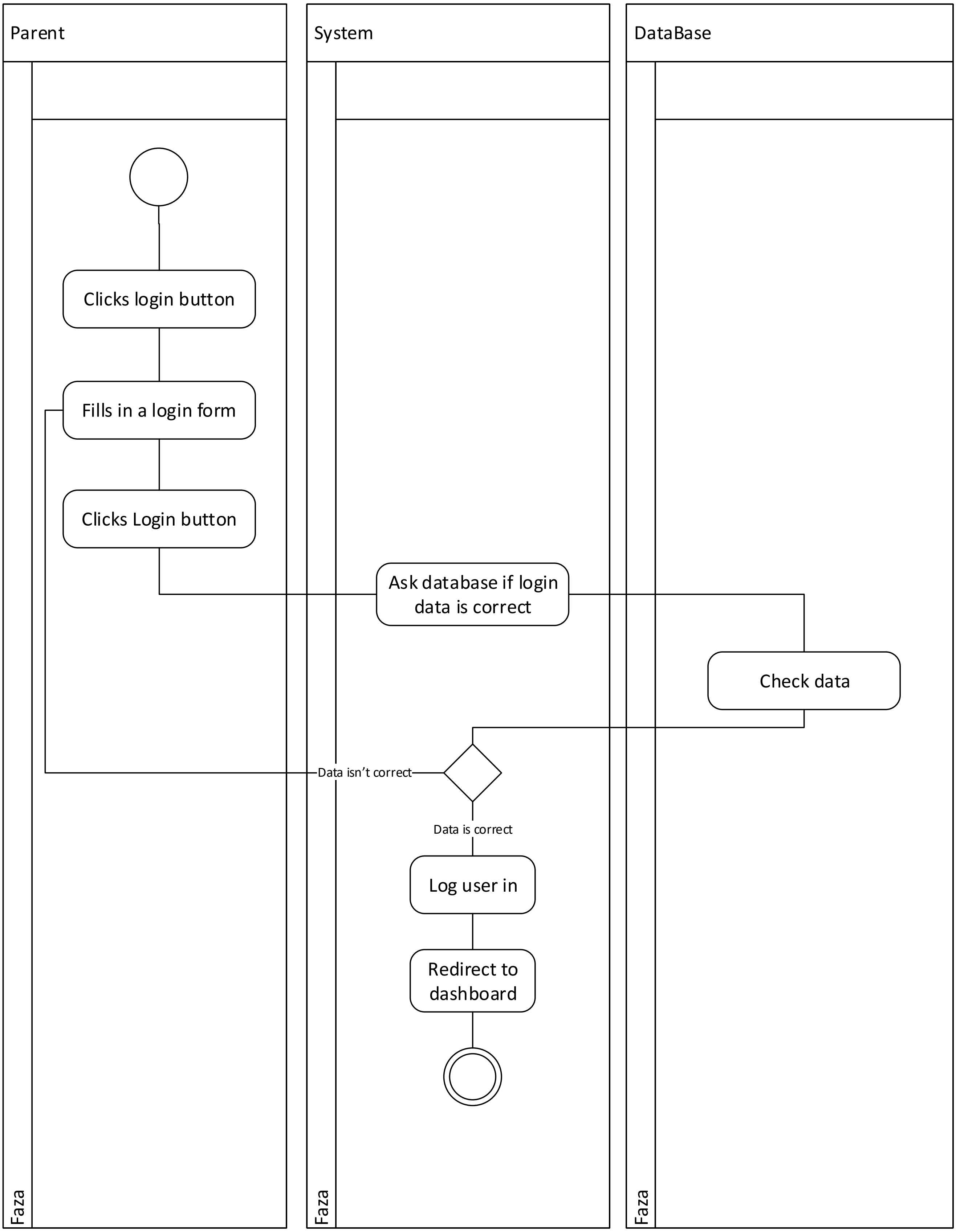
List of capabilities is shown as activity diagrams. These diagrams represents typical actions sequentions. We included both system's and business's activities. In following sections these actions will be presented in form of use cases and scenarios.



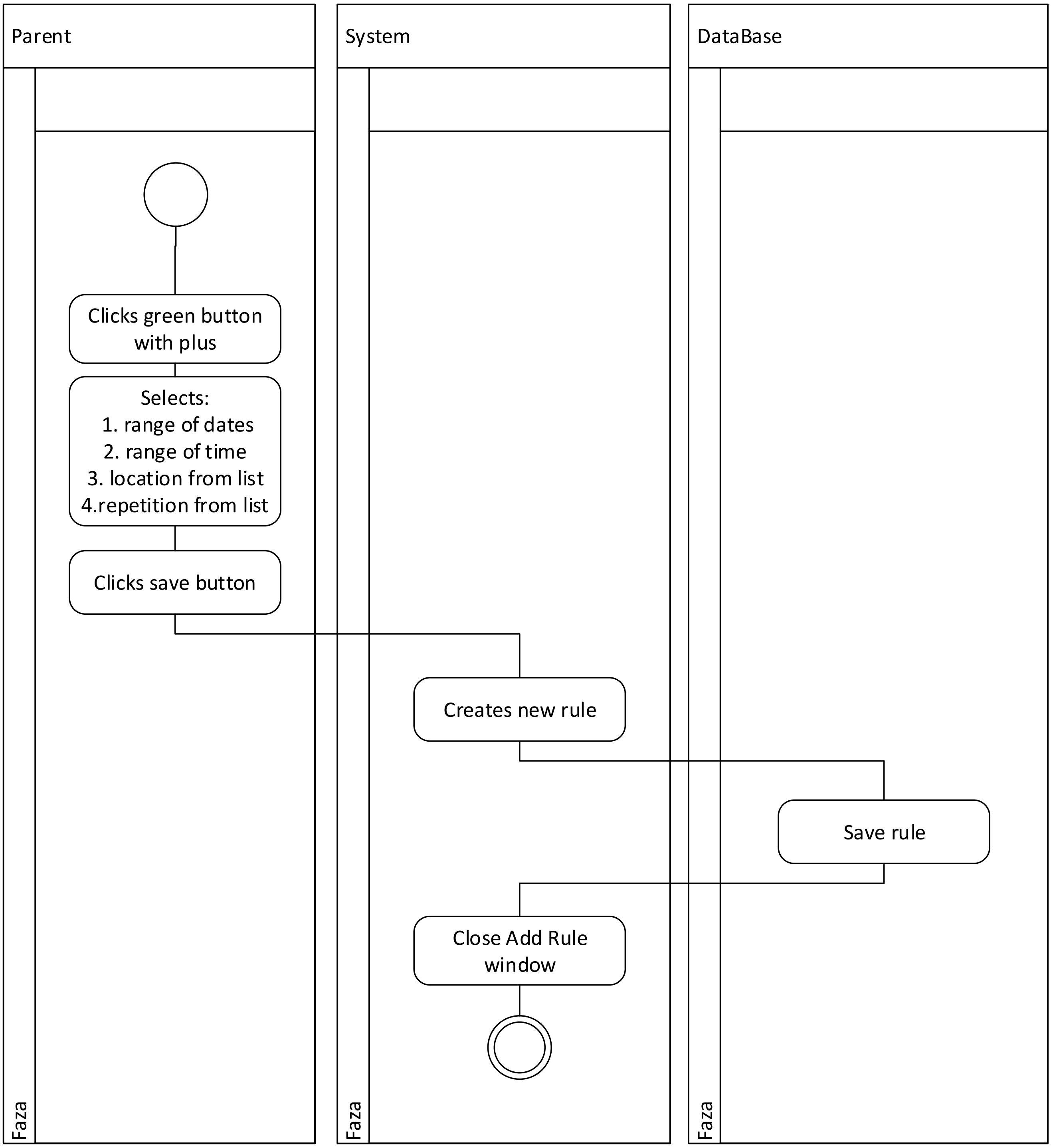
Rysunek 1: Business activity diagram



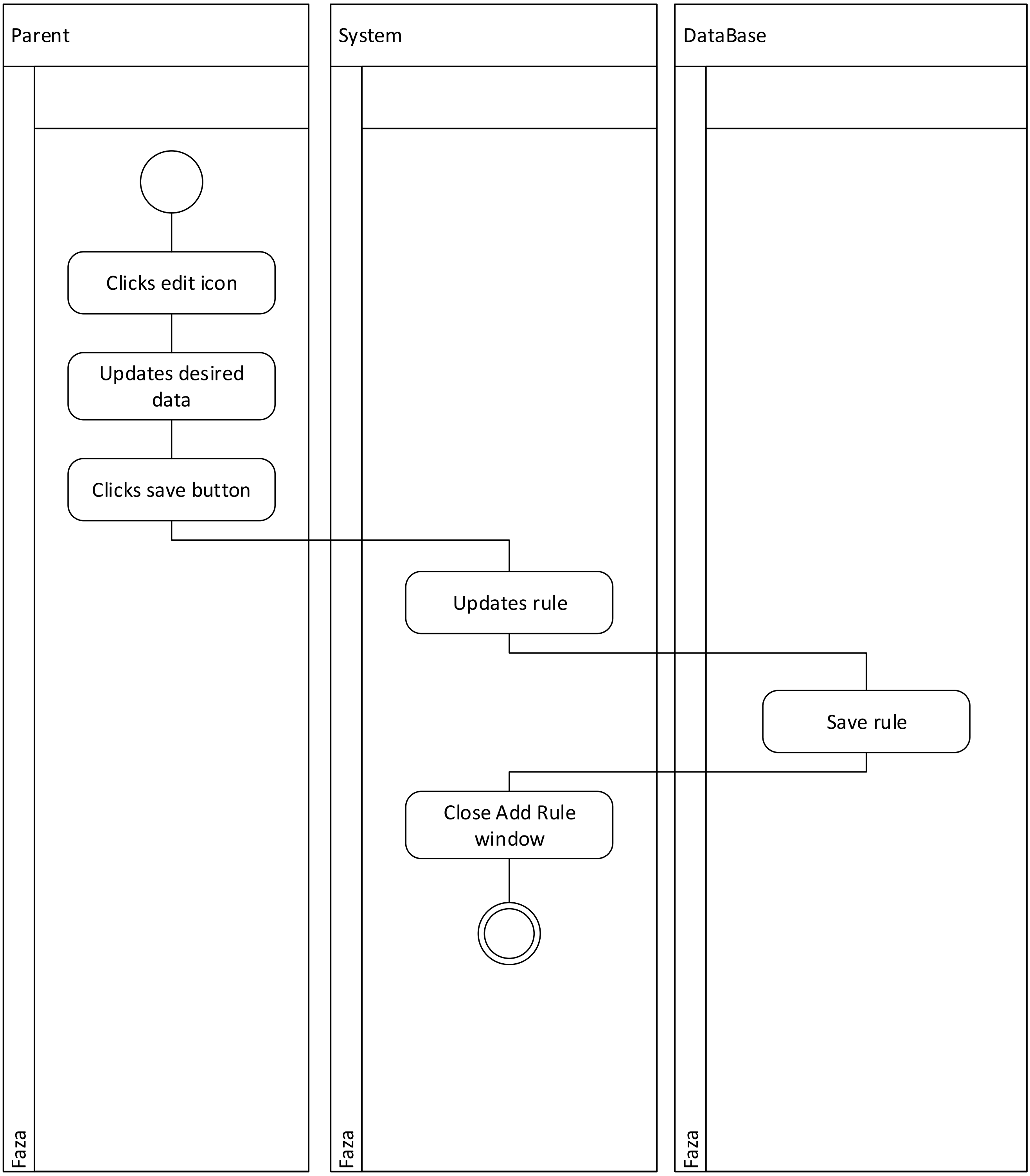
Rysunek 2: Registration activity diagram



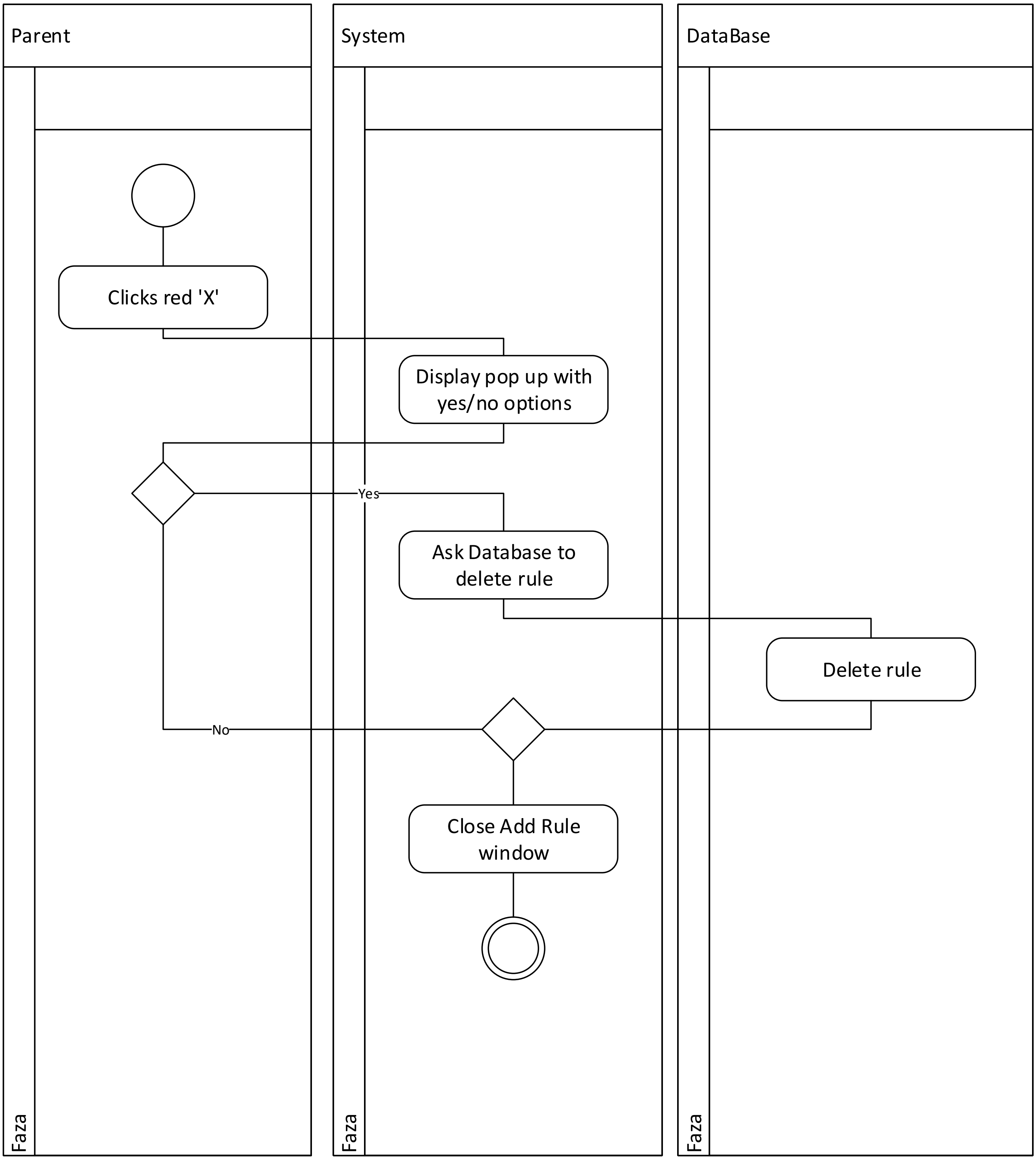
Rysunek 3: Signing in activity diagram



Rysunek 4: Creating new rule activity diagram



Rysunek 5: Updating rule activity diagram



Rysunek 6: Deleting rule activity diagram

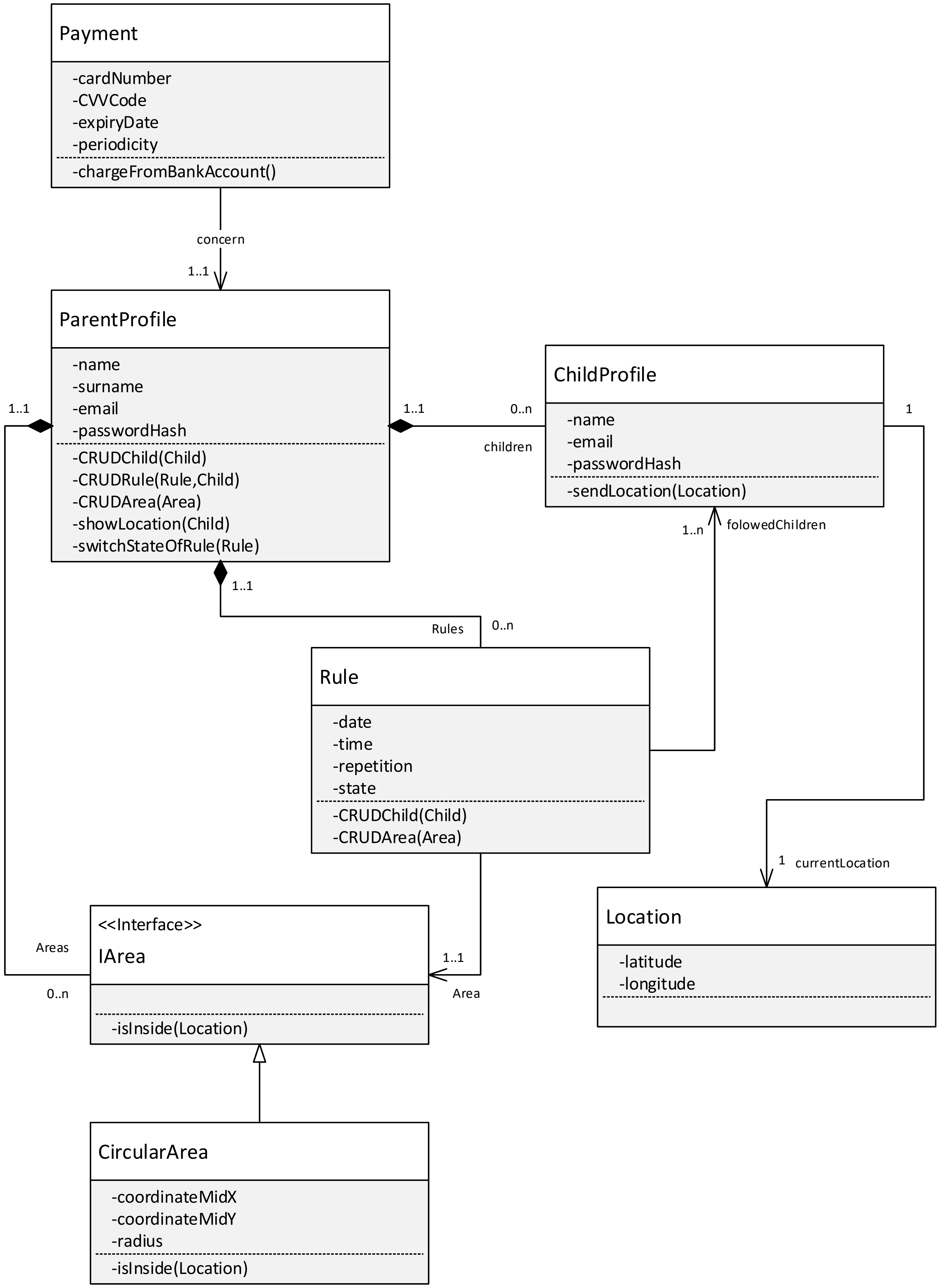
In diagrams we included CRUD operations for rules' management, but didn't include CRUD operations for other components because they are realized in analogous way.

2 Domain's analysis

Classes identified in domain:

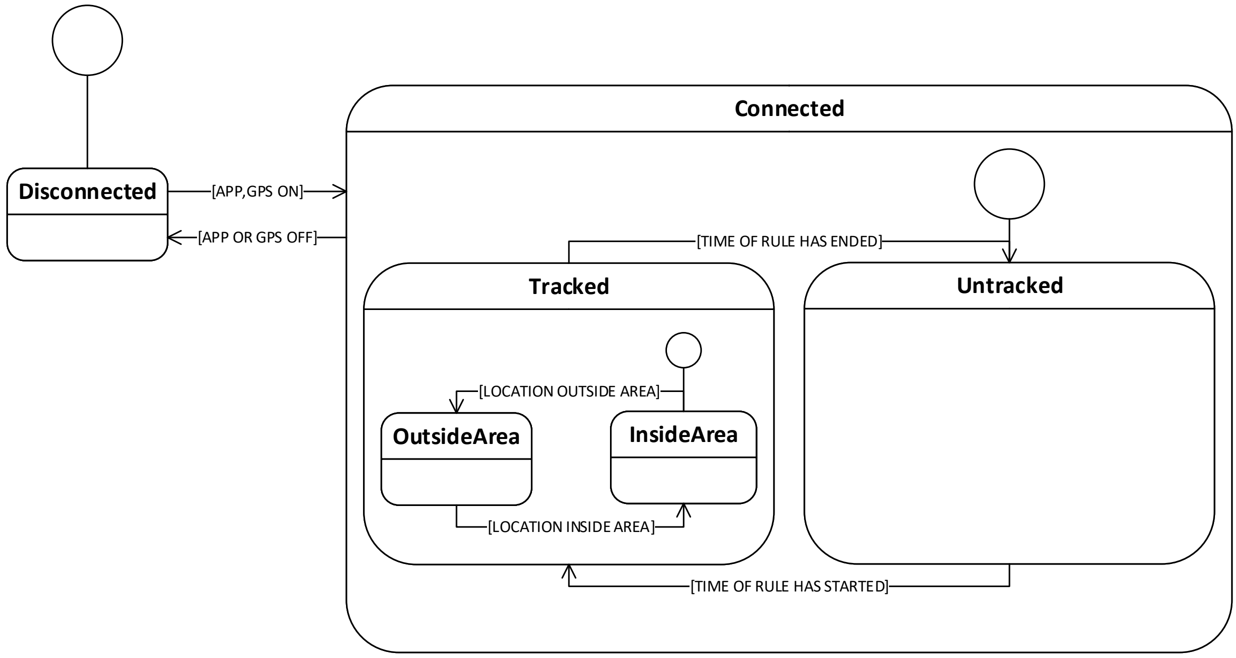
- ParentsAccount - responsible for storing data about parent's account,
- ChildsAccount - responsible for storing data about child's account,
- Rule - responsible for storing data about rules concerning children, area, and time in which rule is active,
- Location - class directly connected with child's account, storing data about current location of child,

- Area Interface - interface for Area's classes,
- CircularArea - class responsible for storing data about center and radius of area.

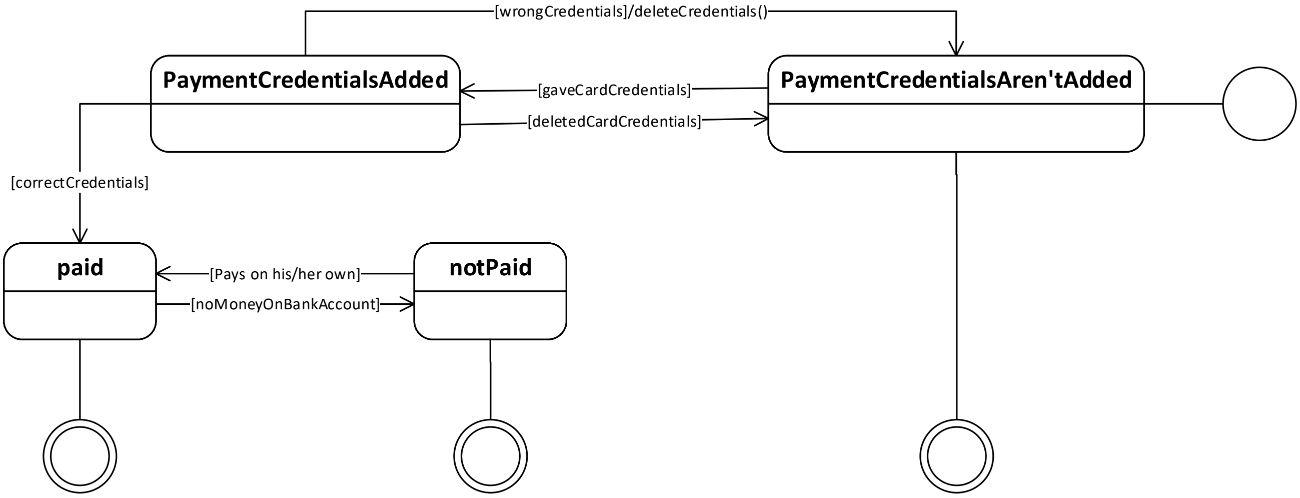


Rysunek 7: Classes diagram

2.1 State diagrams



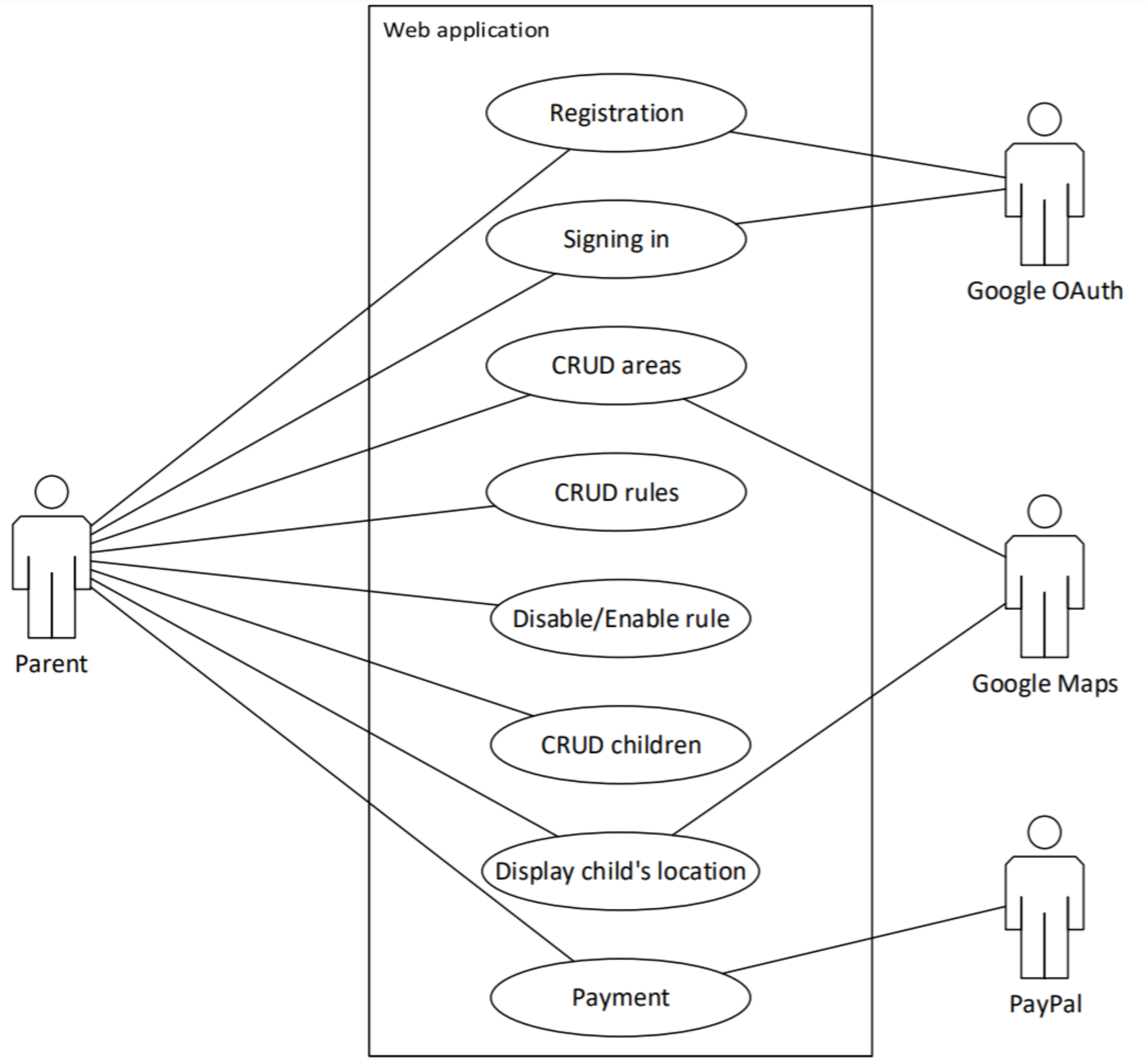
Rysunek 8: State diagram for child's application



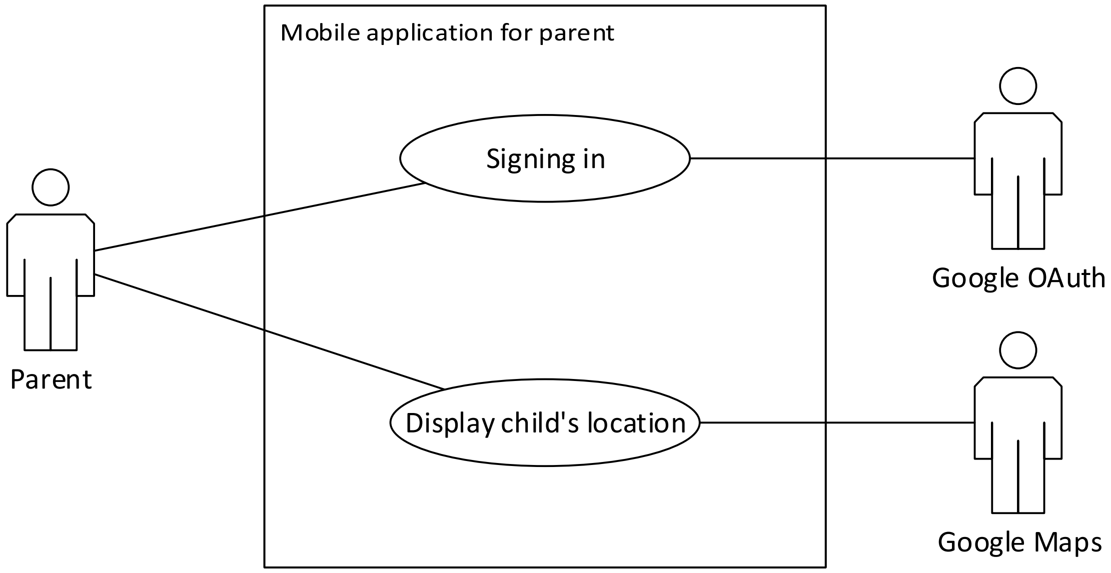
Rysunek 9: State diagram for payment

3 Requirements' specification

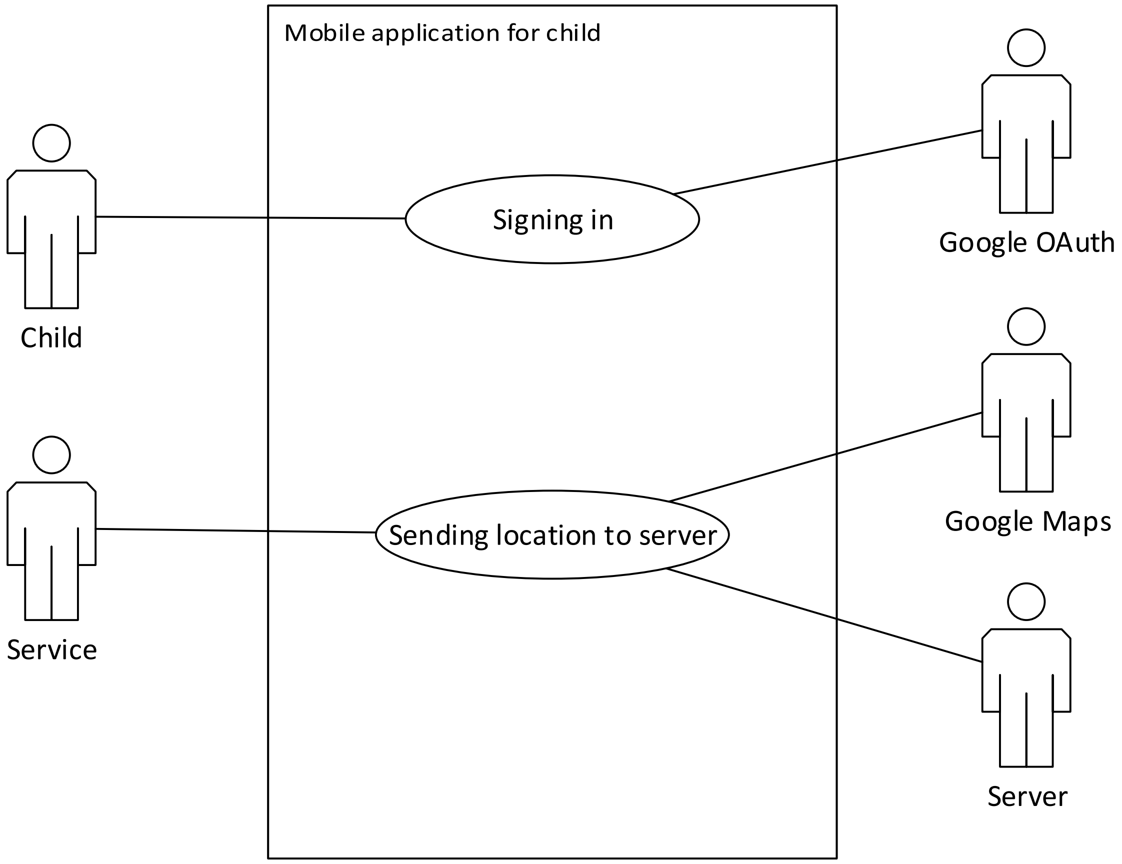
Requirements was shown in form of use cases with scenarios. We aren't showing scenarios for mobile application's use cases because they are too trivial.



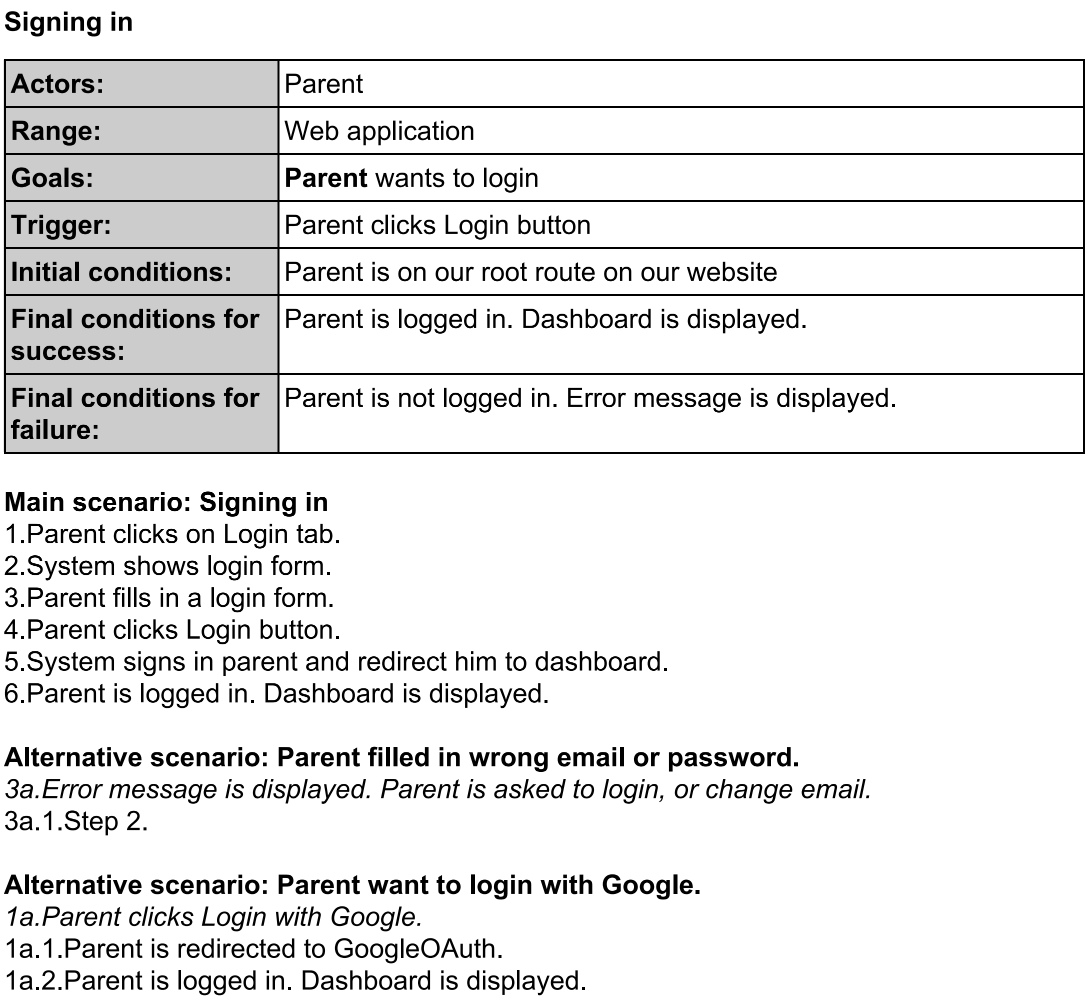
Rysunek 10: Use cases for web application



Rysunek 11: Use cases for parent's mobile application



Rysunek 12: Use cases for child's mobile application



Signing in

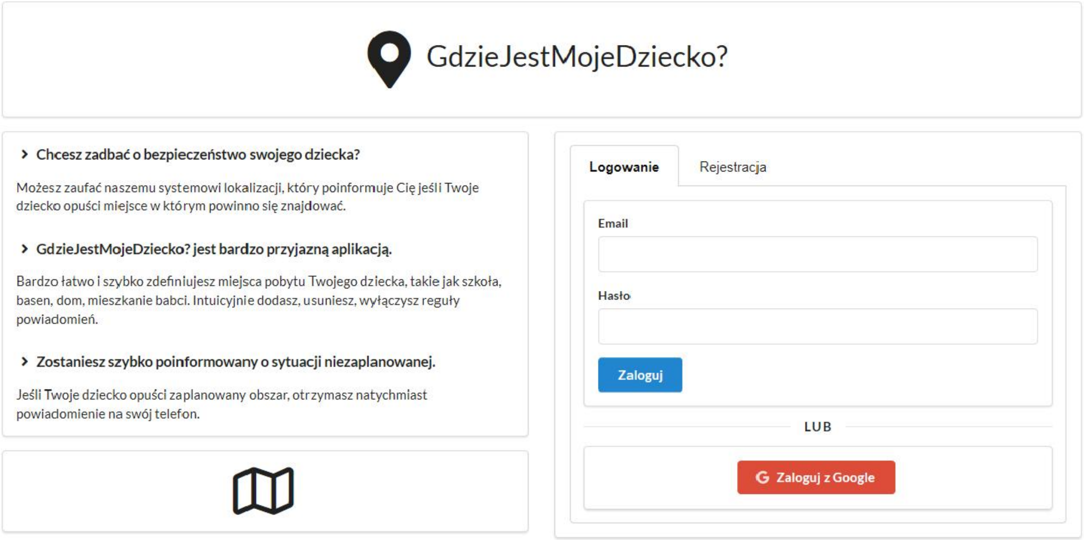
Actors:	Parent
Range:	Web application
Goals:	**Parent** wants to login
Trigger:	Parent clicks Login button
Initial conditions:	Parent is on our root route on our website
Final conditions for success:	Parent is logged in. Dashboard is displayed.
Final conditions for failure:	Parent is not logged in. Error message is displayed.

Main scenario: Signing in
1.Parent clicks on Login tab.
2.System shows login form.
3.Parent fills in a login form.
4.Parent clicks Login button.
5.System signs in parent and redirect him to dashboard.
6.Parent is logged in. Dashboard is displayed.

Alternative scenario: Parent filled in wrong email or password.
3a.Error message is displayed. Parent is asked to login, or change email.
3a.1.Step 2.

Alternative scenario: Parent want to login with Google.
1a.Parent clicks Login with Google.
1a.1.Parent is redirected to GoogleOAuth.
1a.2.Parent is logged in. Dashboard is displayed.

Rysunek 13: Signing in scenario



› Chcesz zadbać o bezpieczeństwo swojego dziecka?

Możesz zaufać naszemu systemowi lokalizacji, który poinformuje Cię jeśli Twoje dziecko opuści miejsce w którym powinno się znajdować.

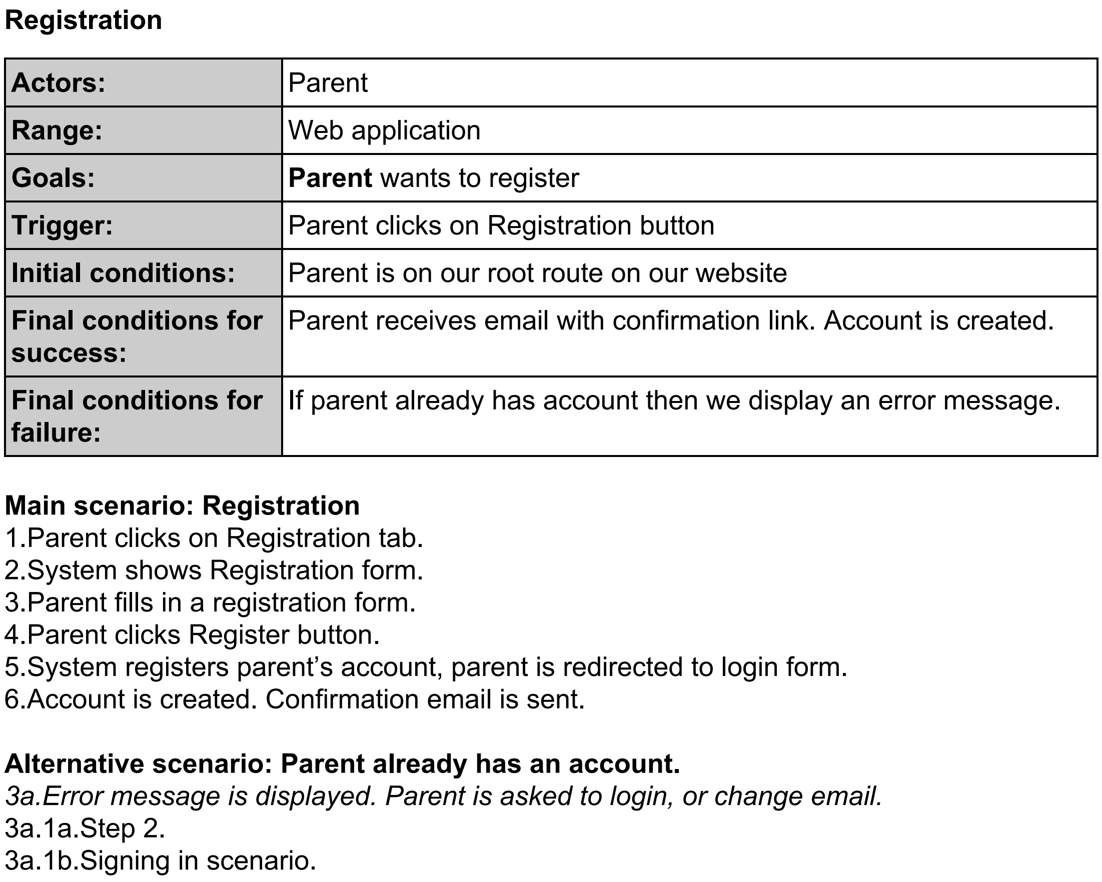
› GdzieJestMojeDziecko? jest bardzo przyjazną aplikacją.

Bardzo łatwo i szybko zdefiniujesz miejsca pobytu Twojego dziecka, takie jak szkoła, basen, dom, mieszkanie babci. Intuicyjnie dodasz, usuniesz, wyłączysz reguły powiadomień.

› Zostaniesz szybko poinformowany o sytuacji niezaplanowanej.

Jeśli Twoje dziecko opuści zaplanowany obszar, otrzymasz natychmiast powiadomienie na swój telefon.

Rysunek 14: Signing in



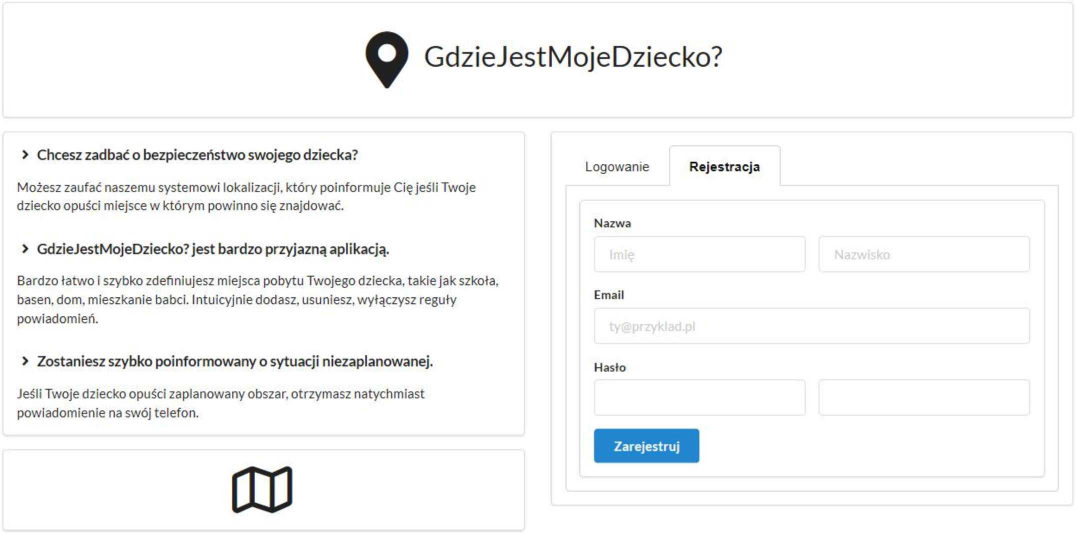
Registration

Actors:	Parent
Range:	Web application
Goals:	**Parent** wants to register
Trigger:	Parent clicks on Registration button
Initial conditions:	Parent is on our root route on our website
Final conditions for success:	Parent receives email with confirmation link. Account is created.
Final conditions for failure:	If parent already has account then we display an error message.

Main scenario: Registration
1.Parent clicks on Registration tab.
2.System shows Registration form.
3.Parent fills in a registration form.
4.Parent clicks Register button.
5.System registers parent's account, parent is redirected to login form.
6.Account is created. Confirmation email is sent.

Alternative scenario: Parent already has an account.
3a.Error message is displayed. Parent is asked to login, or change email.
3a.1a.Step 2.
3a.1b.Signing in scenario.

Rysunek 15: Registration scenario



› **Chcesz zadbać o bezpieczeństwo swojego dziecka?**

Możesz zaufać naszemu systemowi lokalizacji, który poinformuje Cię jeśli Twoje dziecko opuści miejsce w którym powinno się znajdować.

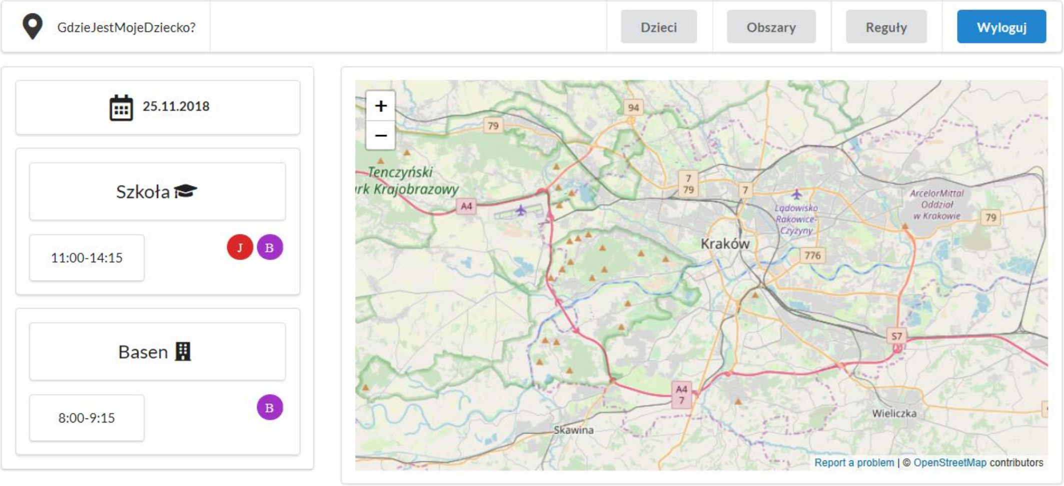
› **GdzieJestMojeDziecko? jest bardzo przyjazną aplikacją.**

Bardzo łatwo i szybko zdefiniujesz miejsca pobytu Twojego dziecka, takie jak szkoła, basen, dom, mieszkanie babci. Intuicyjnie dodasz, usuniesz, wyłączysz reguły powiadomień.

› **Zostaniesz szybko poinformowany o sytuacji niezaplanowanej.**

Jeśli Twoje dziecko opuści zaplanowany obszar, otrzymasz natychmiast powiadomienie na swój telefon.

Rysunek 16: Registering


GdzieJestMojeDziecko?
Dzieci
Obszary
Reguły
Wyloguj
+
−
Tenczyński
rk Krajobrazowy
Kraków
Ładowisko Rakowice-Czyżyny
ArcelorMittal Oddział w Krakowie
Wieliczka
Skawina
Report a problem | © OpenStreetMap contributors

25.11.2018

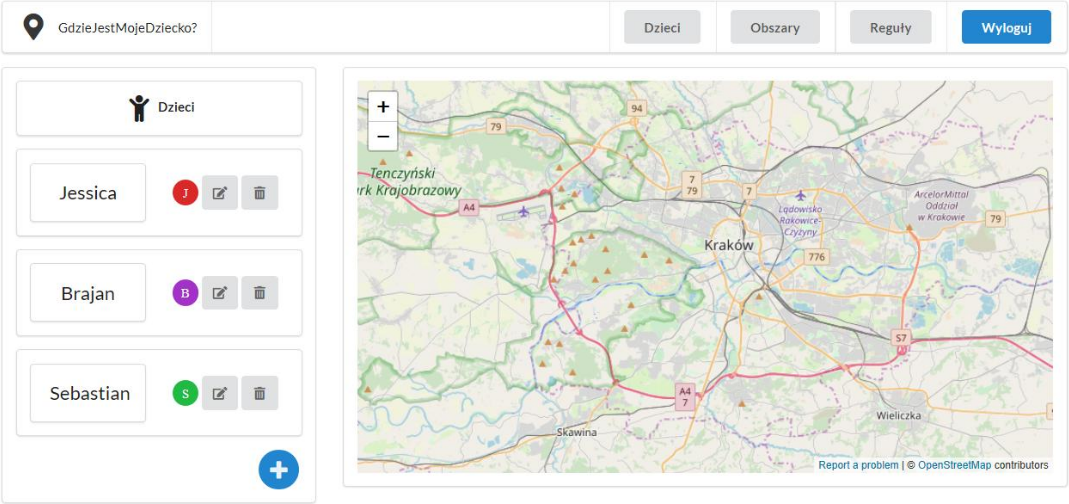
Szkoła

11:00-14:15 J B

Basen

8:00-9:15 B

Rysunek 17: Dashboard


GdzieJestMojeDziecko?
Dzieci
Obszary
Reguły
Wyloguj
+
−
Tenczyński
rk Krajobrazowy
Kraków
Ładowisko Rakowice-Czyżyny
ArcelorMittal Oddział w Krakowie
Wieliczka
Skawina
Report a problem | © OpenStreetMap contributors

Dzieci

Jessica J

Brajan B

Sebastian S

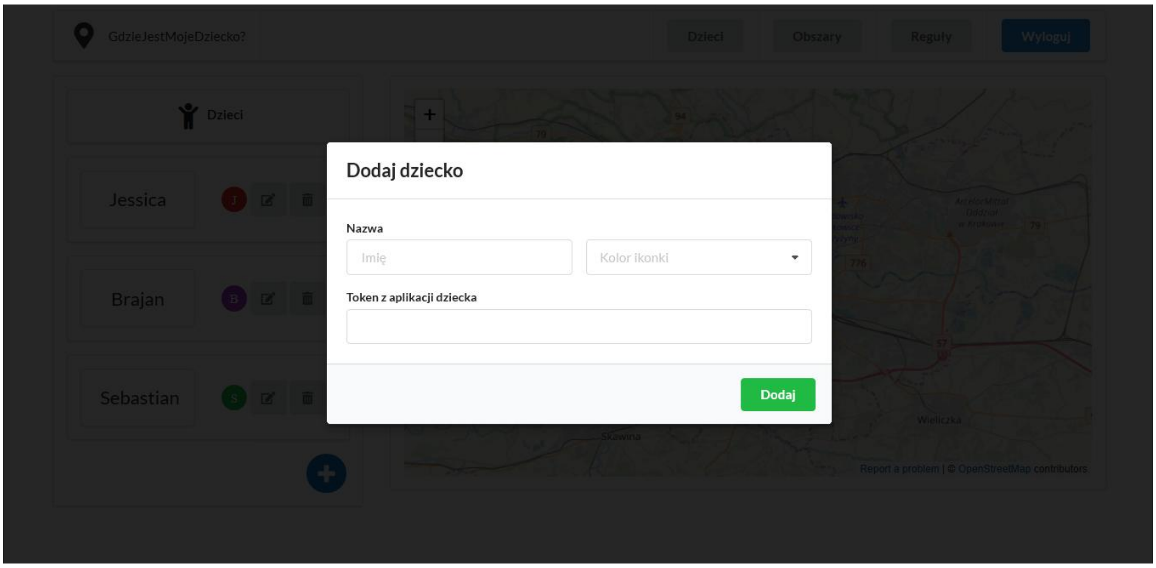
Rysunek 18: Children

CRUD Children: create

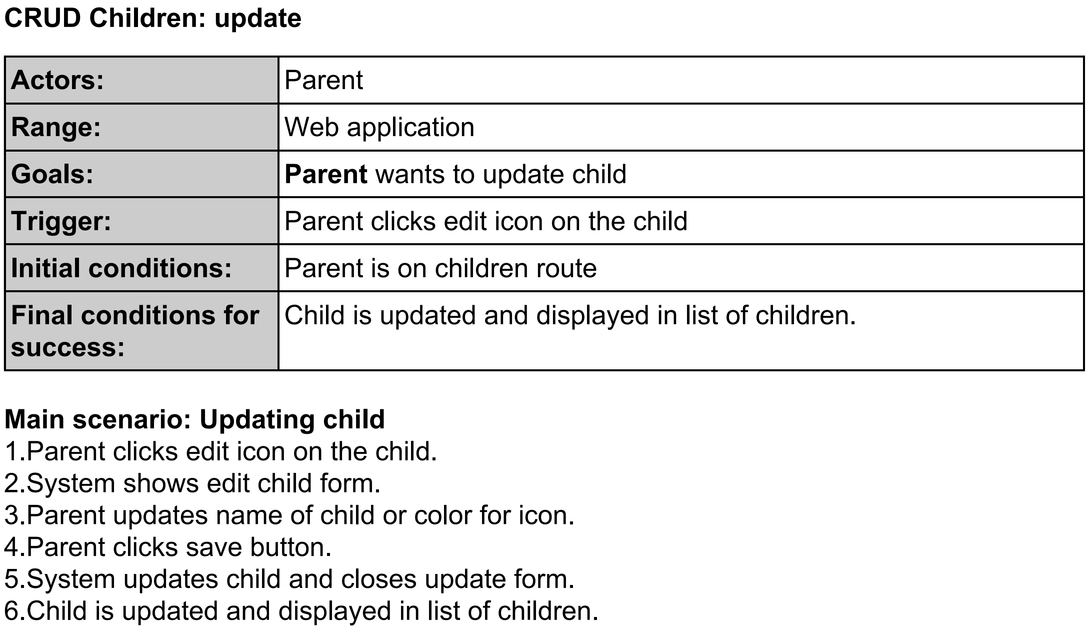
Actors:	Parent
Range:	Web application
Goals:	**Parent** wants to create child
Trigger:	Parent clicks blue button with plus
Initial conditions:	Parent is on children route
Final conditions for success:	Child is created and displayed in list of children

Main scenario: Creating child
1.Parent clicks blue button with plus.
2.System shows form for creating child.
3.Parent types name of child.
4.Parent chooses color for icon of child.
5.Parent clicks save button.
6.System creates child and closes form.
7.Child is created and displayed in list of children.

Rysunek 19: Creating child scenario



Rysunek 20: Creating child

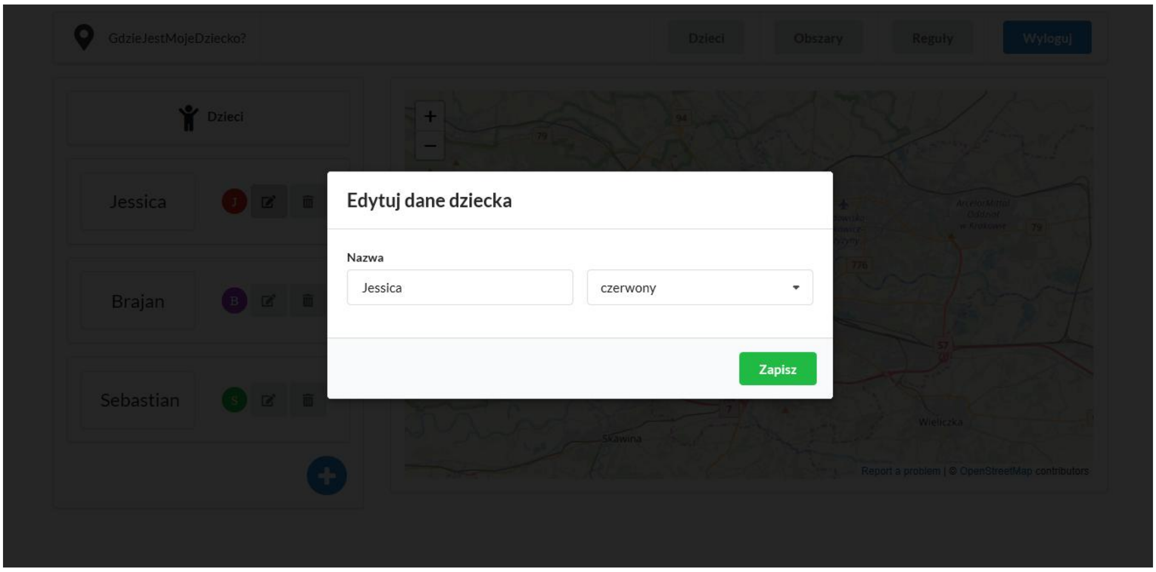


CRUD Children: update

Actors:	Parent
Range:	Web application
Goals:	**Parent** wants to update child
Trigger:	Parent clicks edit icon on the child
Initial conditions:	Parent is on children route
Final conditions for success:	Child is updated and displayed in list of children.

Main scenario: Updating child
1.Parent clicks edit icon on the child.
2.System shows edit child form.
3.Parent updates name of child or color for icon.
4.Parent clicks save button.
5.System updates child and closes update form.
6.Child is updated and displayed in list of children.

Rysunek 21: Updating child scenario



Edytuj dane dziecka

Nazwa: Jessica

czerwony

Rysunek 22: Editing child data

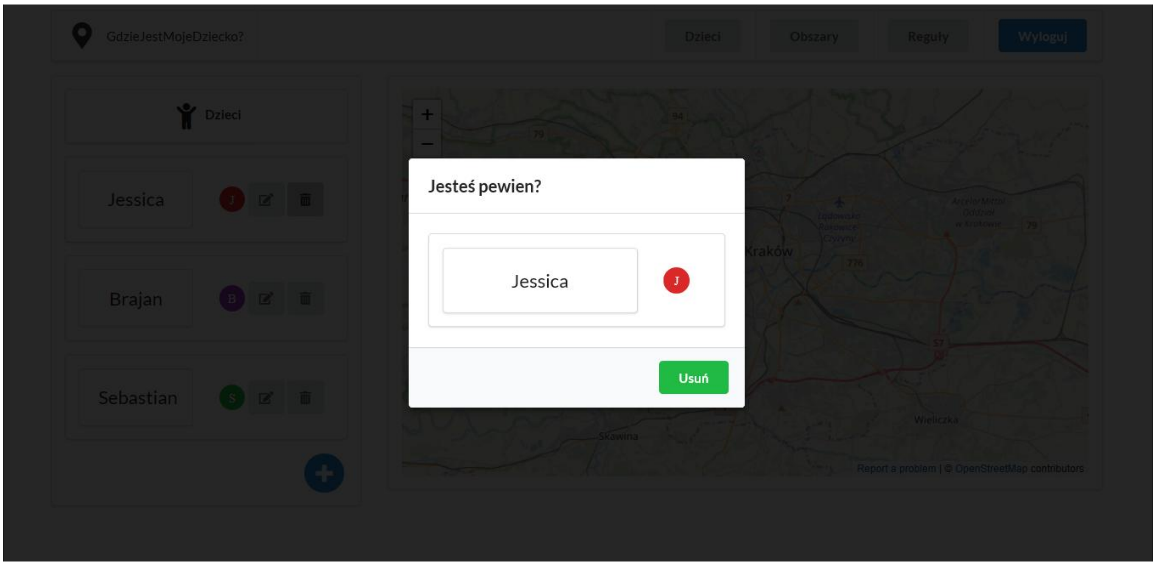
CRUD Children: delete

Actors:	Parent
Range:	Web application
Goals:	**Parent** wants to delete child
Trigger:	Parent clicks trash icon on the child
Initial conditions:	Parent is on children route
Final conditions for success:	Child is deleted and list of children is updated.
Final conditions for failure:	Child isn't deleted.

Main scenario: Deleting child
1.Parent clicks trash icon on the child.
2.Pop up is displayed with question "Are you sure?" with "Yes" button.
3.Parent clicks "Yes".
4.Child is deleted and displayed and list of children is updated.

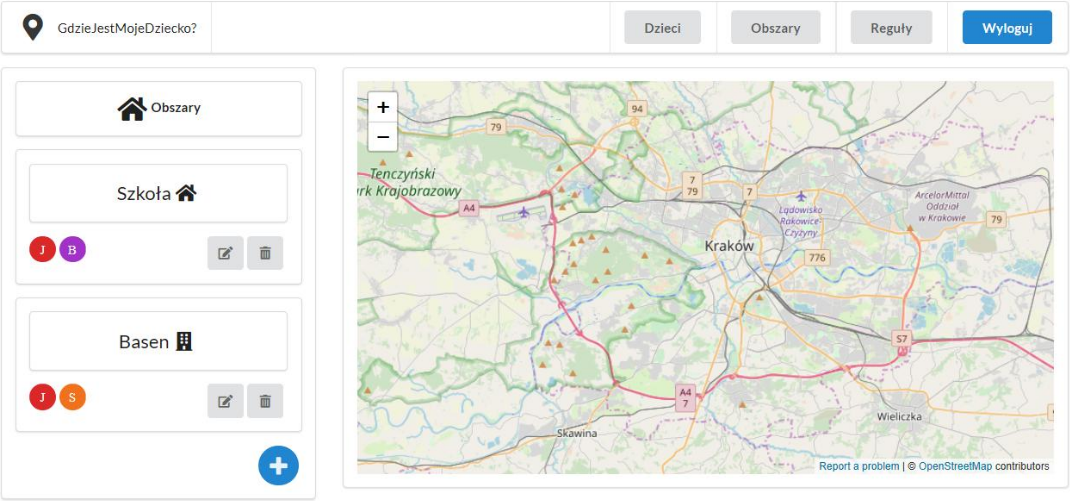
Alternative scenario: Parent doesn't want to delete.
3a.Parent clicks outside of popup.
3a.1.Child isn't deleted.

Rysunek 23: Deleting child scenario



Jesteś pewien?

Rysunek 24: Deleting child



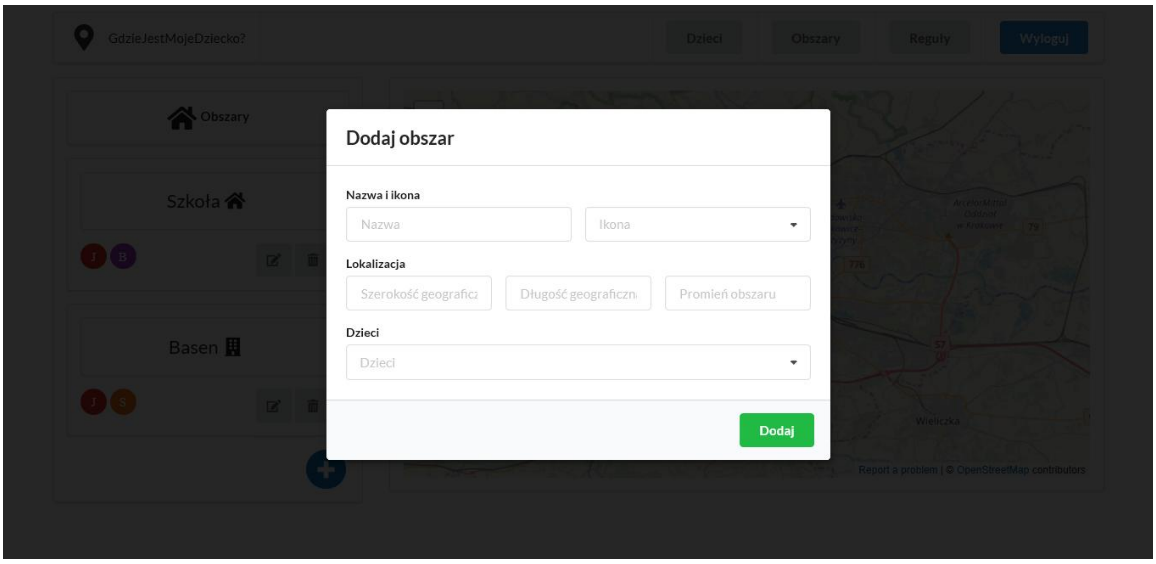
Rysunek 25: Areas

CRUD Areas: create

Actors:	Parent
Range:	Web application
Goals:	**Parent** wants to create area
Trigger:	Parent clicks blue button with plus
Initial conditions:	Parent is on areas route
Final conditions for success:	Area is created and displayed in list of areas.

Main scenario: Creating area
1.Parent clicks blue button with plus.
2.System shows form for creating area.
3.Parent types name of area and chooses icon for area.
4.Parent types address of location.
5.Parent sets radius of area.
6.Parent adds child/children.
7.Parent clicks save button.
8.System creates area and closes form.
9.Area is created and displayed in list of areas.

Rysunek 26: Creating area scenario



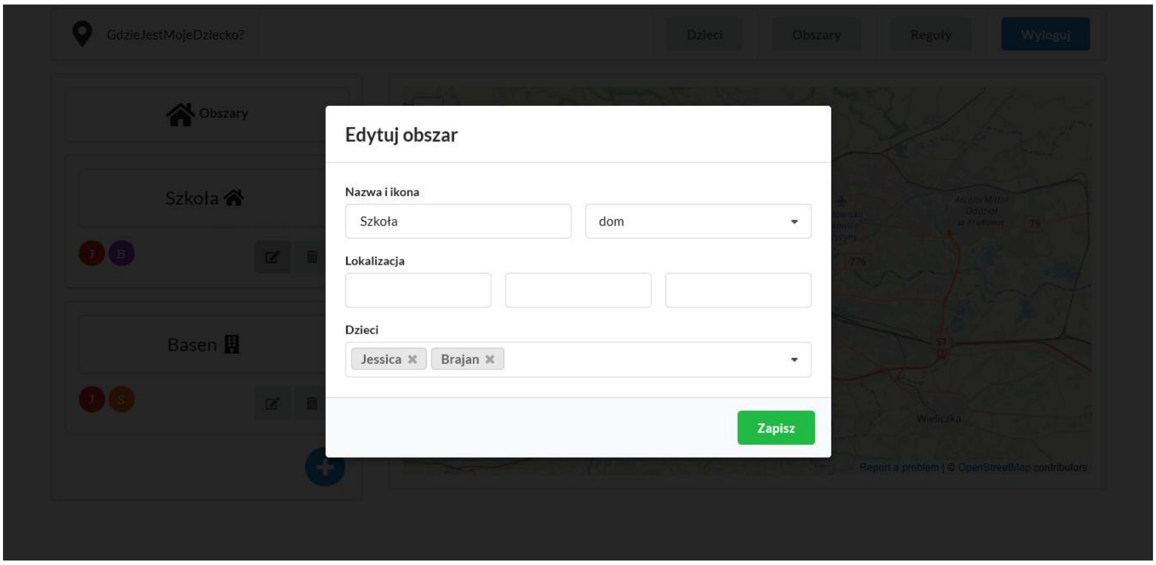
Rysunek 27: Creating area

CRUD Areas: update

Actors:	Parent
Range:	Web application
Goals:	**Parent** wants to update area
Trigger:	Parent clicks edit icon on the area
Initial conditions:	Parent is on areas route
Final conditions for success:	Area is updated and displayed in list of areas.

Main scenario: Updating area
1.Parent clicks edit icon on the area.
2.System shows editing form.
3.Parent updates name of area or location address or radius or list of children.
4.Parent clicks save button.
5.System updates area and closes form.
6.Area is updated and displayed in list of areas.

Rysunek 28: Updating area scenario



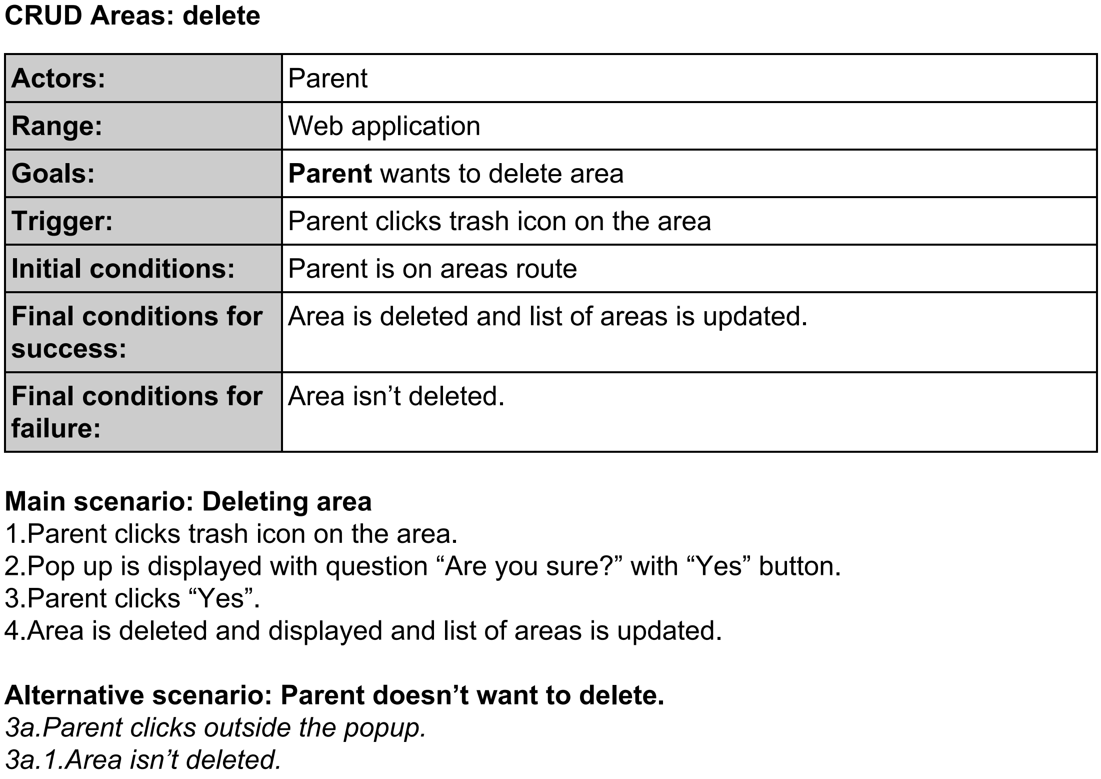
Edytuj obszar

Nazwa i ikona: Szkoła — dom

Lokalizacja

Dzieci: Jessica, Brajan

Rysunek 29: Editing area



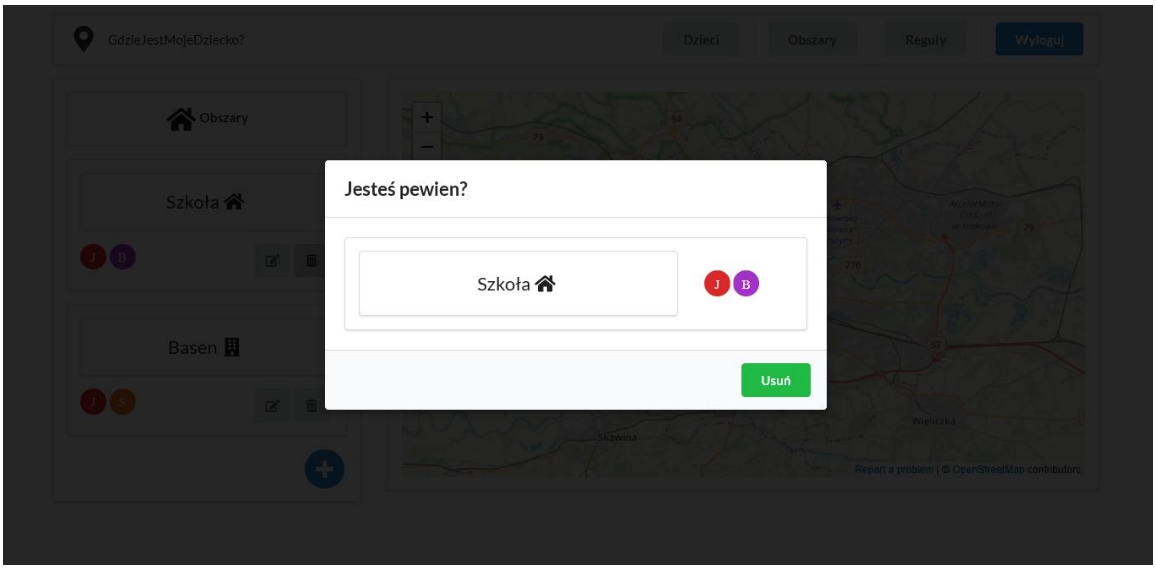
CRUD Areas: delete

Actors:	Parent
Range:	Web application
Goals:	**Parent** wants to delete area
Trigger:	Parent clicks trash icon on the area
Initial conditions:	Parent is on areas route
Final conditions for success:	Area is deleted and list of areas is updated.
Final conditions for failure:	Area isn’t deleted.

Main scenario: Deleting area
1.Parent clicks trash icon on the area.
2.Pop up is displayed with question “Are you sure?” with “Yes” button.
3.Parent clicks “Yes”.
4.Area is deleted and displayed and list of areas is updated.

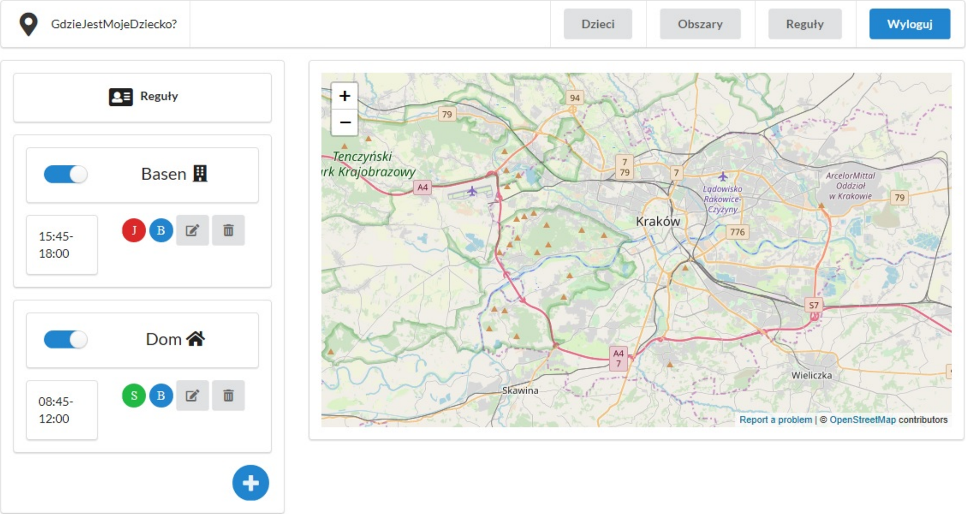
Alternative scenario: Parent doesn’t want to delete.
3a.Parent clicks outside the popup.
3a.1.Area isn’t deleted.

Rysunek 30: Deleting area scenario

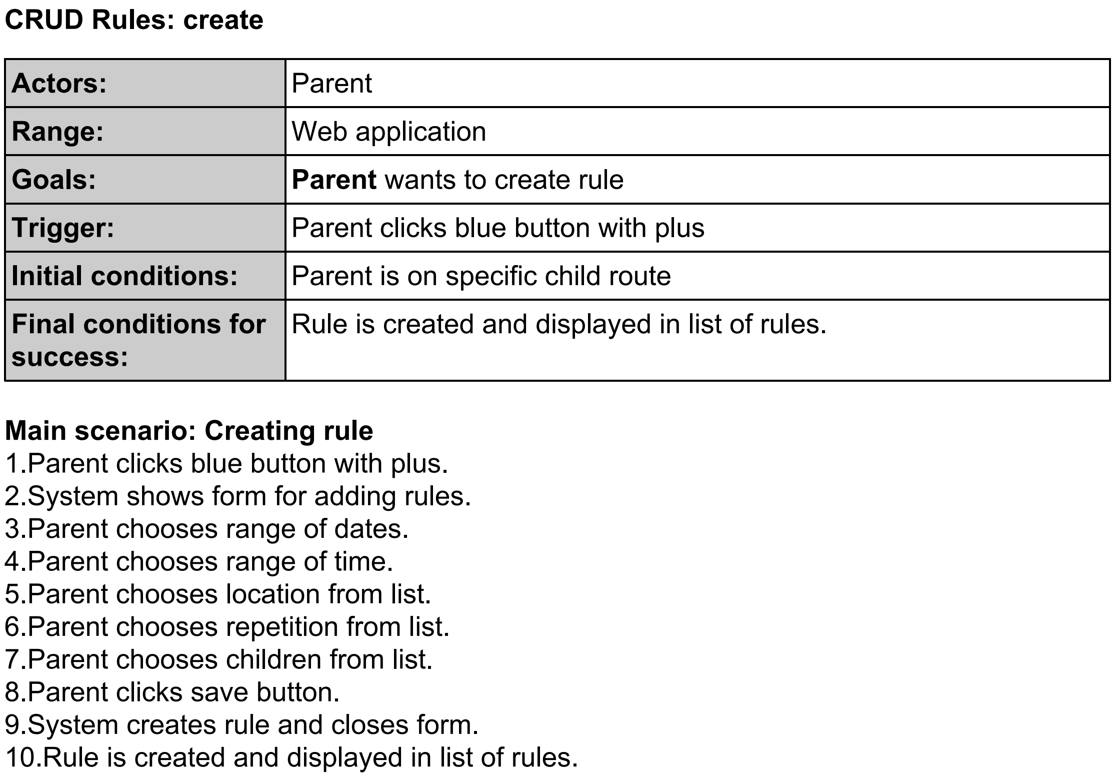


Jesteś pewien?

Rysunek 31: Deleting area



Rysunek 32: Rules



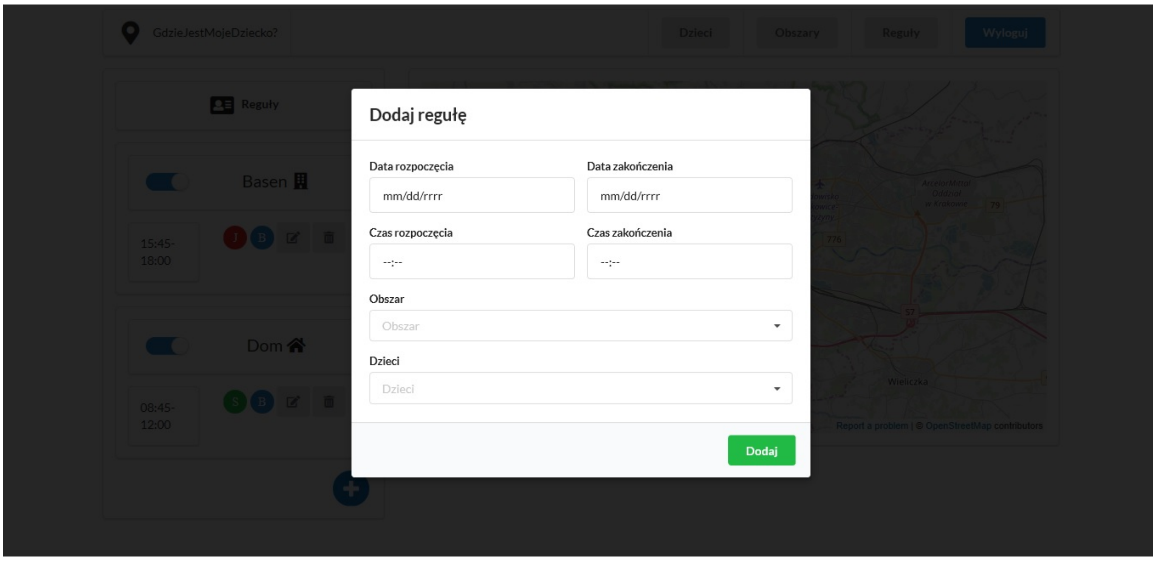
CRUD Rules: create

Actors:	Parent
Range:	Web application
Goals:	**Parent** wants to create rule
Trigger:	Parent clicks blue button with plus
Initial conditions:	Parent is on specific child route
Final conditions for success:	Rule is created and displayed in list of rules.

Main scenario: Creating rule

1.Parent clicks blue button with plus.
2.System shows form for adding rules.
3.Parent chooses range of dates.
4.Parent chooses range of time.
5.Parent chooses location from list.
6.Parent chooses repetition from list.
7.Parent chooses children from list.
8.Parent clicks save button.
9.System creates rule and closes form.
10.Rule is created and displayed in list of rules.

Rysunek 33: Creating rule scenario



Dodaj regułę

Data rozpoczęcia: mm/dd/rrrr
Data zakończenia: mm/dd/rrrr
Czas rozpoczęcia: --:--
Czas zakończenia: --:--
Obszar: Obszar
Dzieci: Dzieci

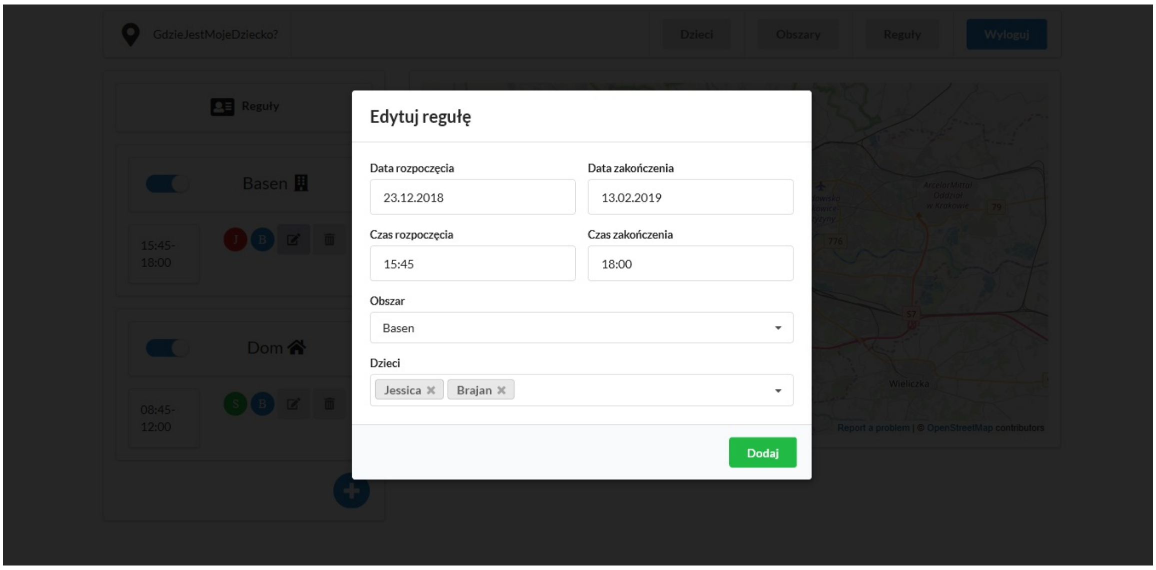
Rysunek 34: Creating rule

CRUD Rules: update

Actors:	Parent
Range:	Web application
Goals:	**Parent** wants to update rule
Trigger:	Parent clicks edit icon on the rule
Initial conditions:	Parent is on rules route
Final conditions for success:	Rule is updated and displayed in list of rules.

Main scenario: Updating rule
1.Parent clicks edit icon on the rule.
2.System shows edit rule form.
3.Parent updates range of dates or range of time or location or repetition.
4.Parent clicks save button.
5.System updates rule and closes form.
6.Rule is updated and displayed in list of rules.

Rysunek 35: Updating rule scenario



Edytuj regułę

Data rozpoczęcia: 23.12.2018

Data zakończenia: 13.02.2019

Czas rozpoczęcia: 15:45

Czas zakończenia: 18:00

Obszar: Basen

Dzieci: Jessica, Brajan

Rysunek 36: Updating rule

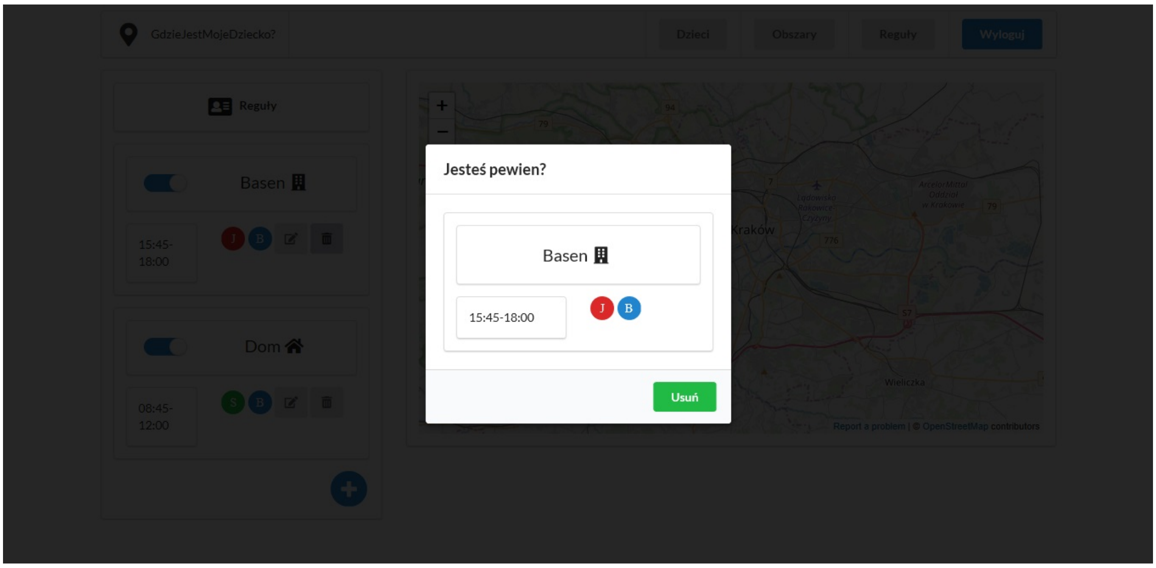
CRUD Rules: delete

Actors:	Parent
Range:	Web application
Goals:	**Parent** wants to delete rule
Trigger:	Parent clicks trash icon on the rule
Initial conditions:	Parent is on rules route
Final conditions for success:	Rule is deleted and list of rules is updated.
Final conditions for failure:	Rule isn't deleted.

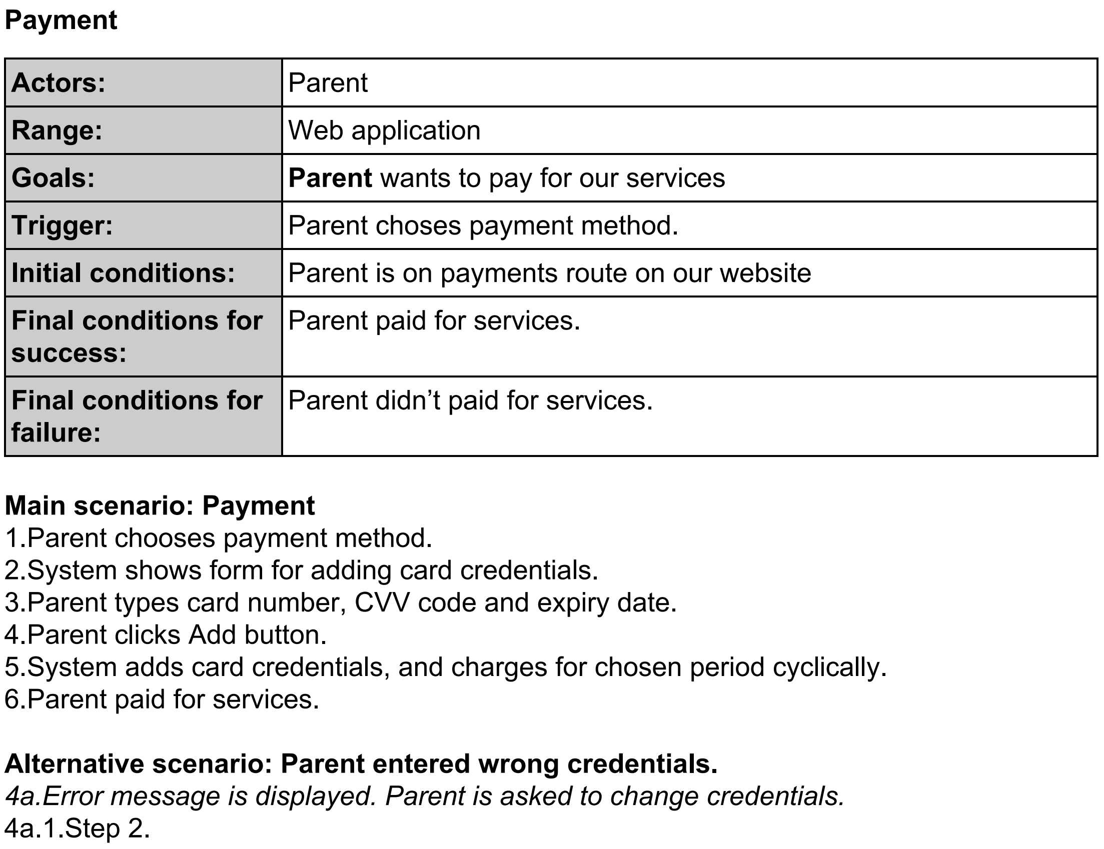
Main scenario: Deleting rule
1.Parent clicks trash icon on the rule.
2.Pop up is displayed with question "Are you sure?" with "Yes" button.
3.Parent clicks "Yes".
4.Rule is deleted and displayed and list of rules is updated.

Alternative scenario: Parent doesn't want to delete.
3a.Parent clicks outside of popup.
*3a.1.*Rule *isn't deleted.*

Rysunek 37: Deleting rule scenario



Rysunek 38: Deleting rule



Payment

Actors:	Parent
Range:	Web application
Goals:	**Parent** wants to pay for our services
Trigger:	Parent choses payment method.
Initial conditions:	Parent is on payments route on our website
Final conditions for success:	Parent paid for services.
Final conditions for failure:	Parent didn’t paid for services.

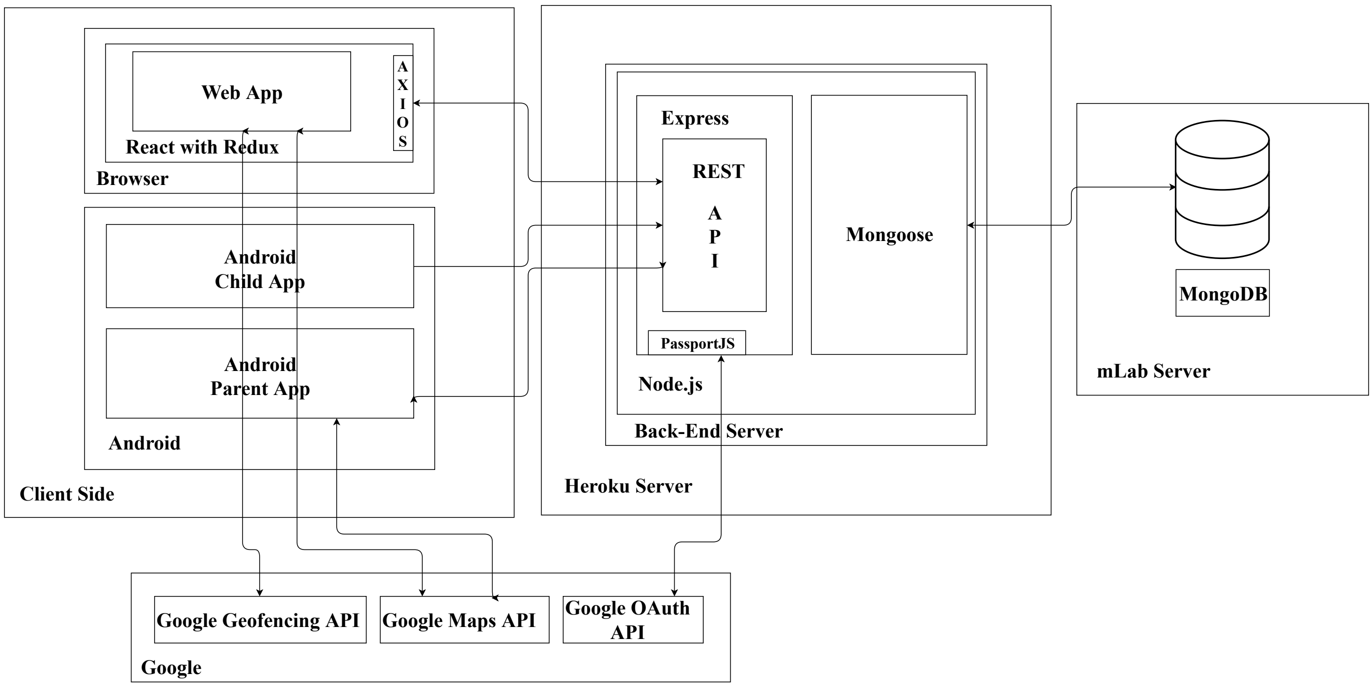
Main scenario: Payment
1.Parent chooses payment method.
2.System shows form for adding card credentials.
3.Parent types card number, CVV code and expiry date.
4.Parent clicks Add button.
5.System adds card credentials, and charges for chosen period cyclically.
6.Parent paid for services.

Alternative scenario: Parent entered wrong credentials.
4a.Error message is displayed. Parent is asked to change credentials.
4a.1.Step 2.

Rysunek 39: Payment scenario

4 System’s architecture

In our application we are using MongoDB, so our database is identical to our class diagram.



Rysunek 40: Architecture, modules and interfaces

4.1 Project of tests

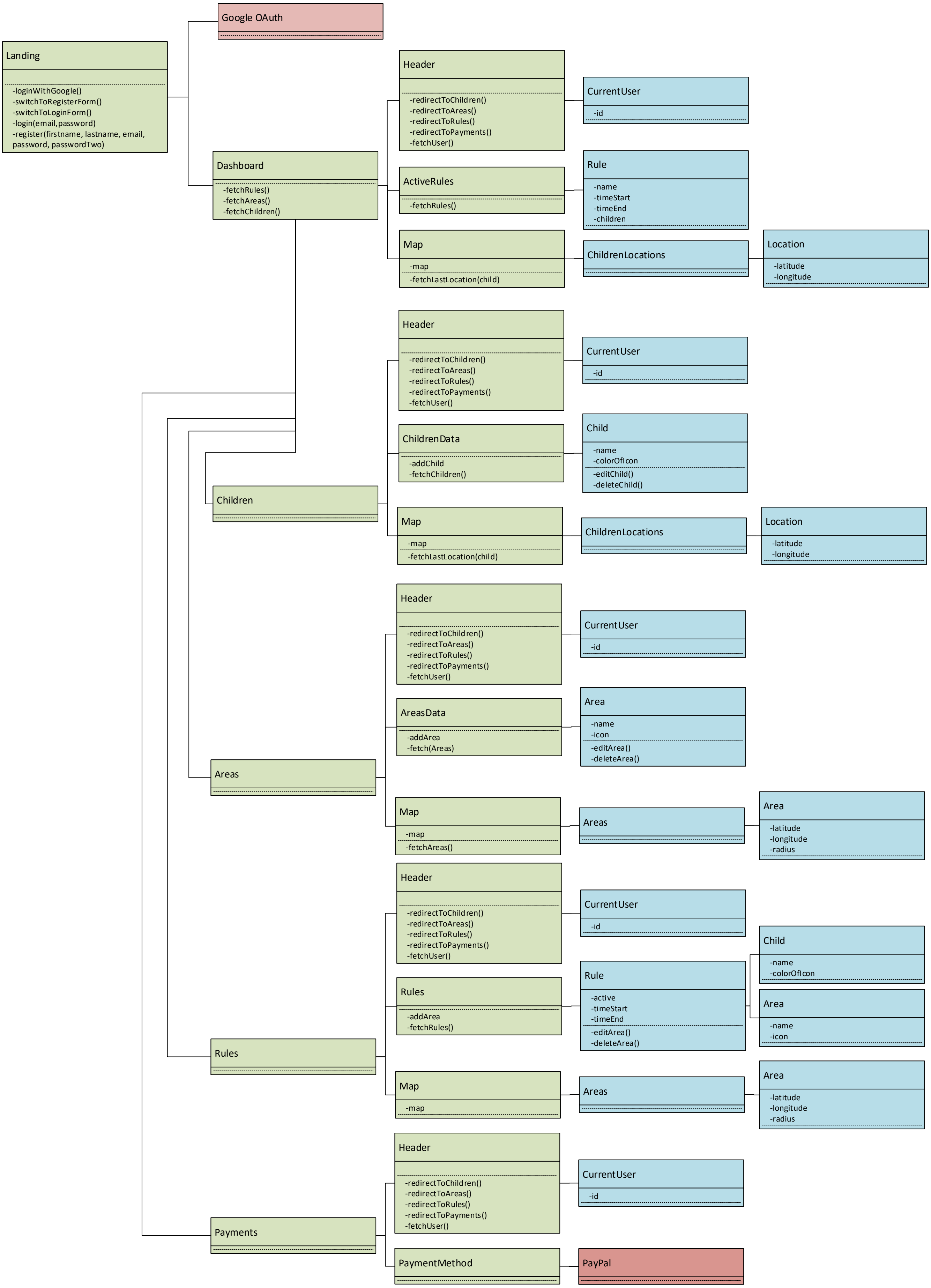
Tests	**What?**	**Technologies**
Unit	backend and frontend methods	Jasmine
Integration	communication between modules	Jasmine
Functional	use cases	Jasmine, Selenium
Manual	Mobile application	manually
Manual	API	postman

Tablica 2: Project of tests

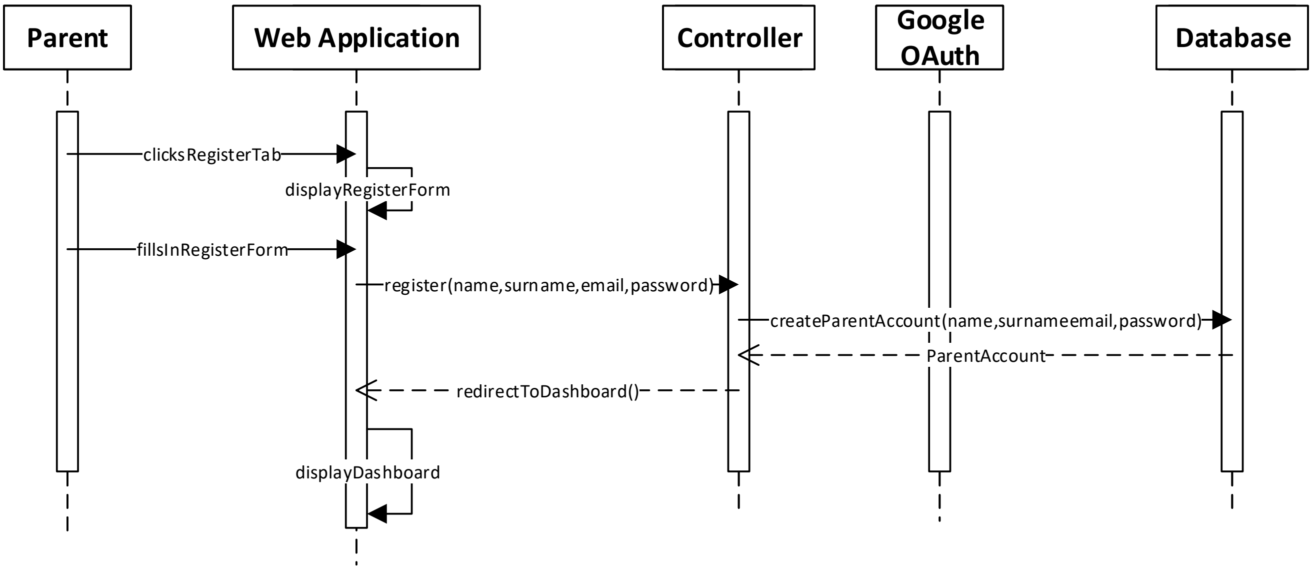
5 Project of software

5.1 Client side: React Components

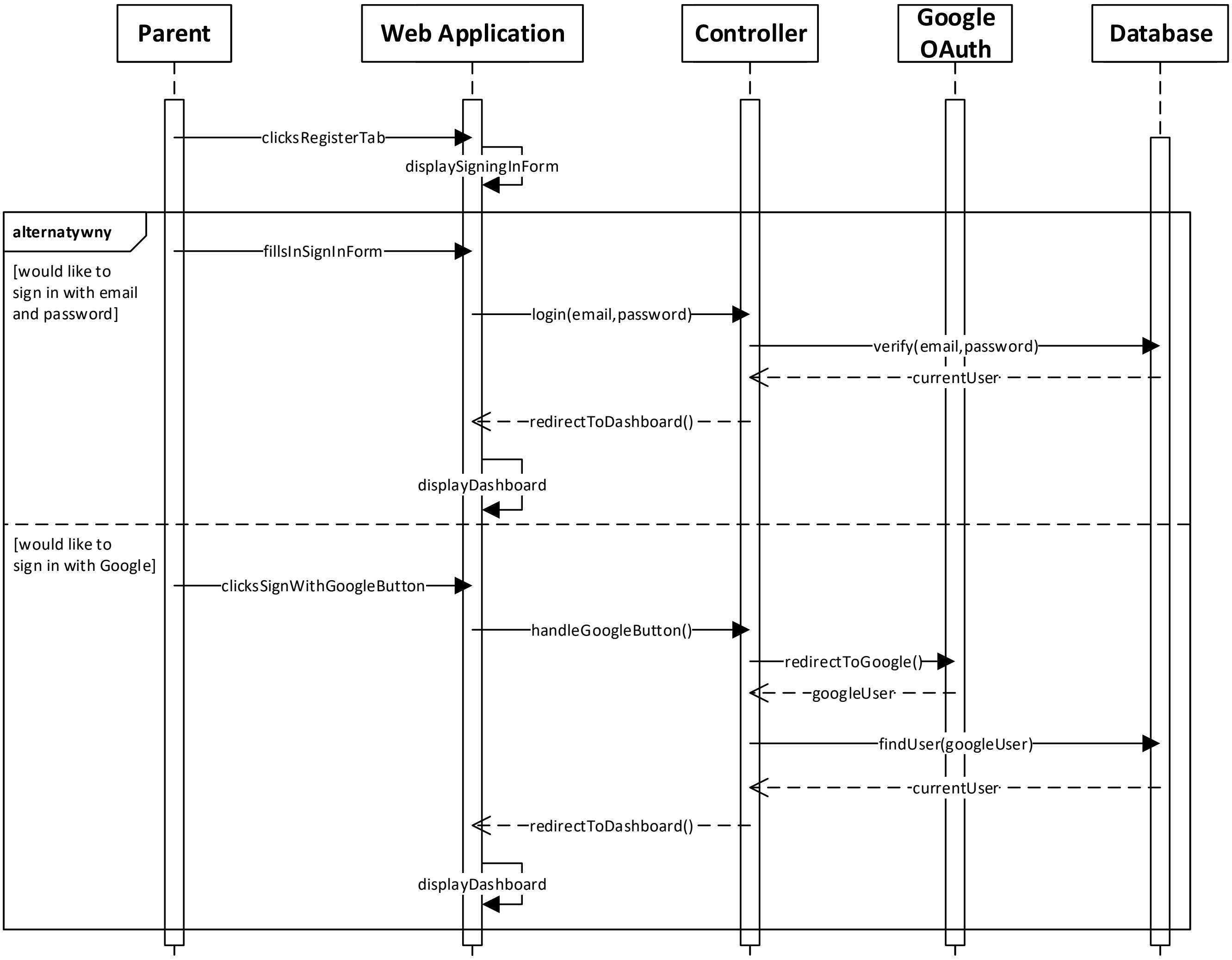
Platform: JavaScript -> JSX -> HTML



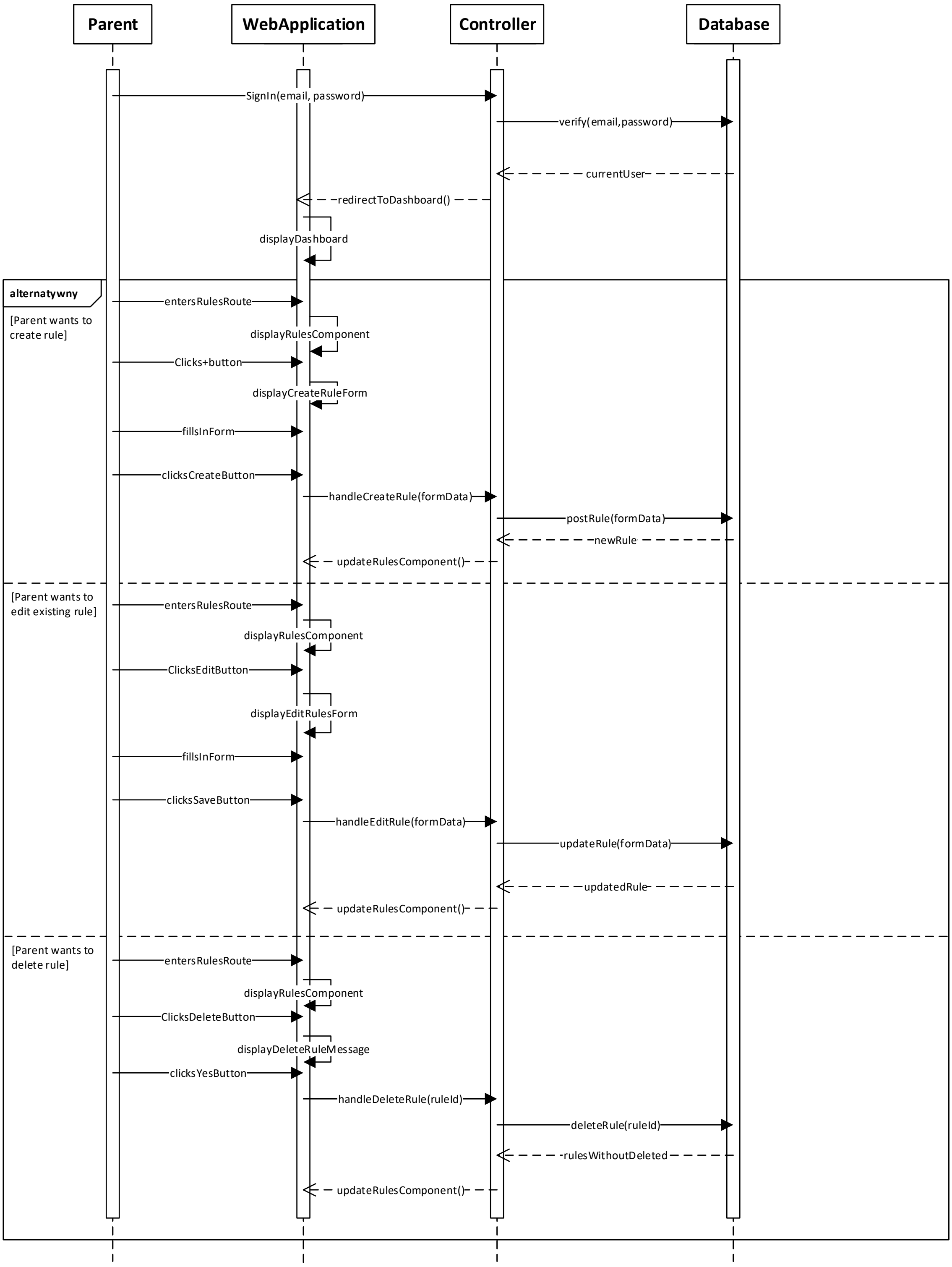
Rysunek 41: Software project



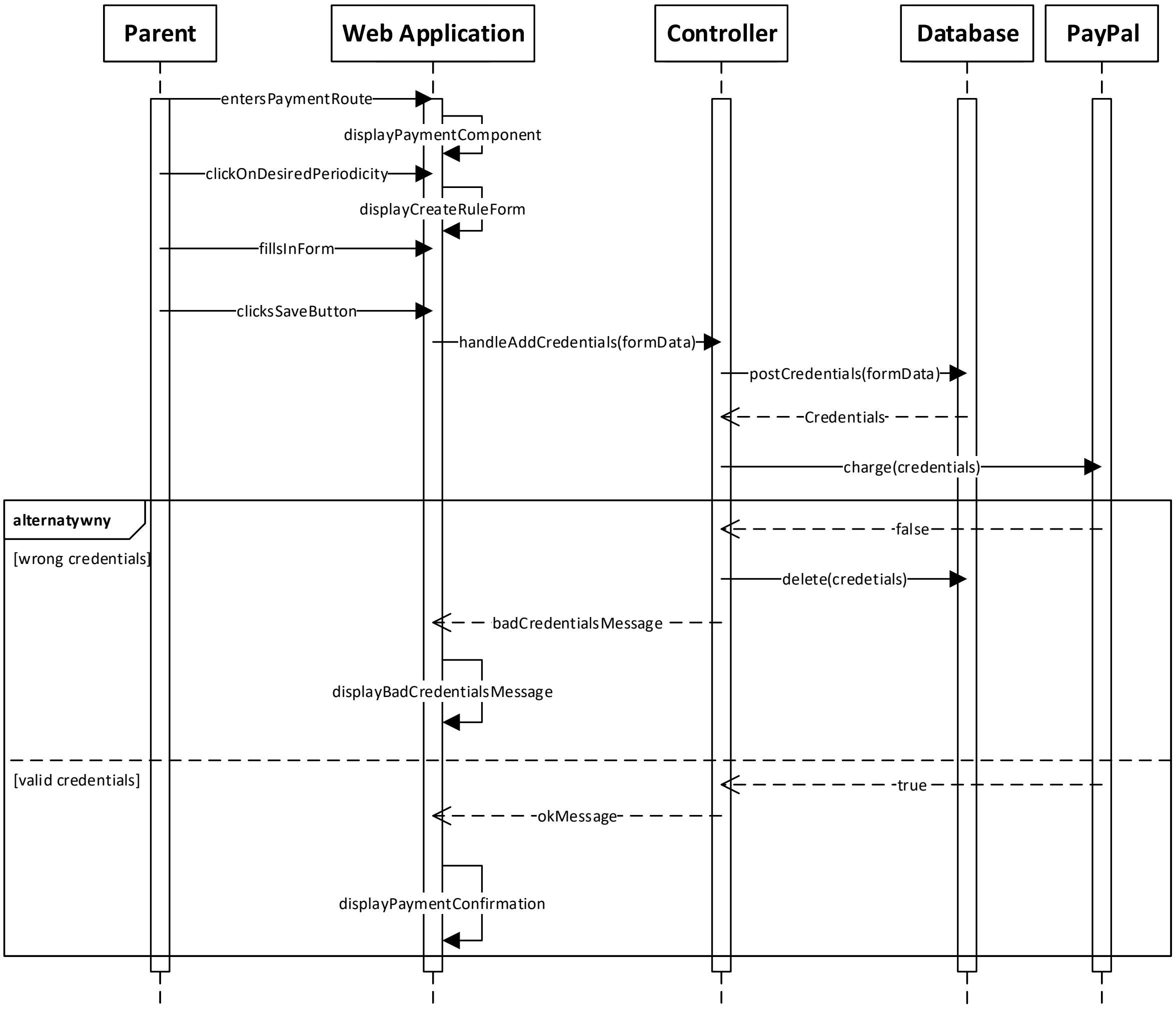
Rysunek 42: Register Sequence Diagram



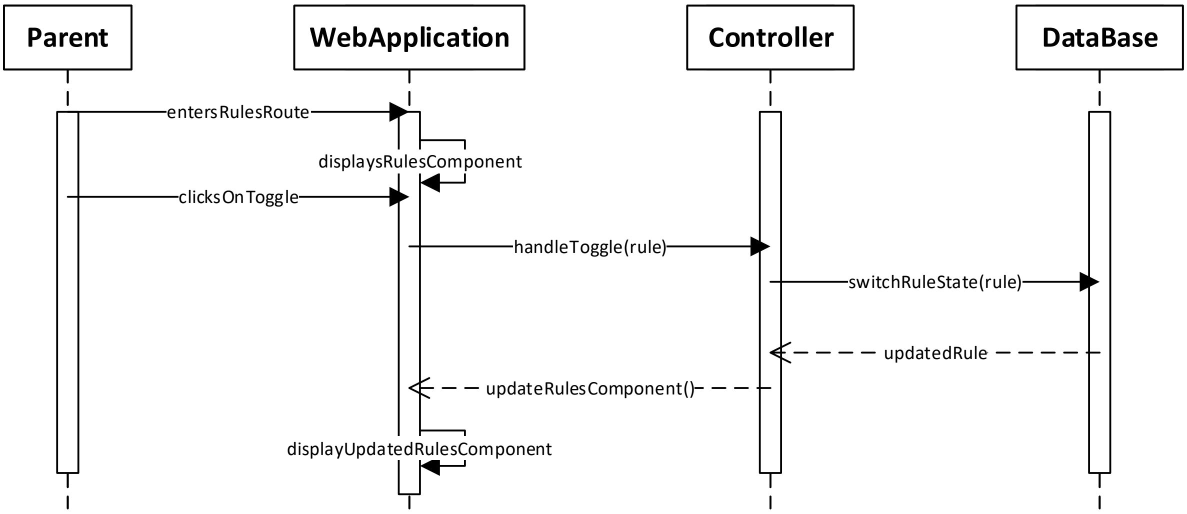
Rysunek 43: Login Sequence Diagram



Rysunek 44: CRUD Rules Sequence Diagram



Rysunek 45: Payment Rules Sequence Diagram



Rysunek 46: Toogle Rule's State Sequence Diagram

5.2 Server side: API

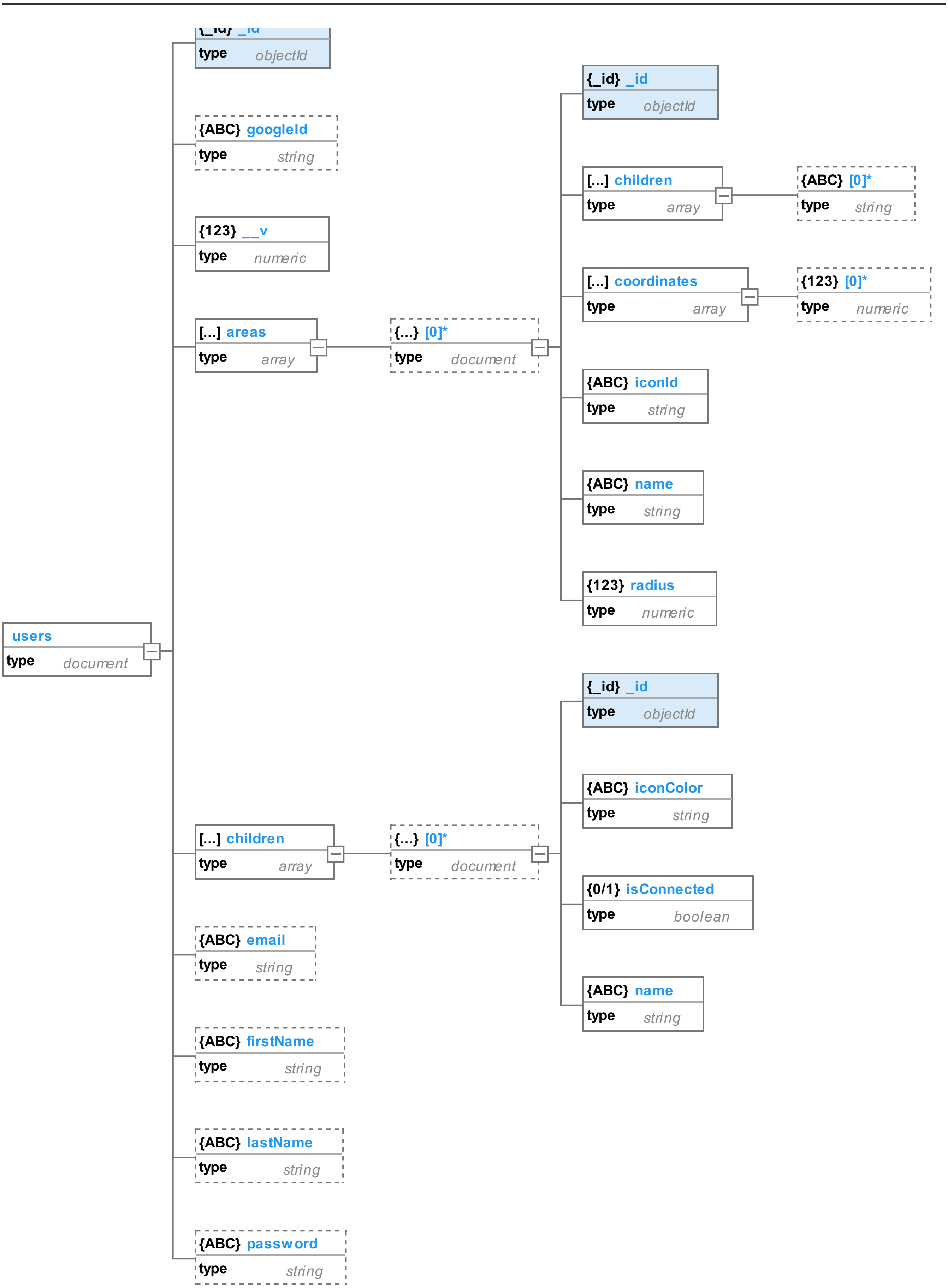
Platform: JavaScript -> NodeJS -> Express

Our application's modules communicate with server through RESTful API. Our application will send requests to server when user fills forms, and clicks buttons. Server will fetch proper data using axios from MongoDB database on mLab server. Based on fetched data web components will update or user will be redirected to proper route.

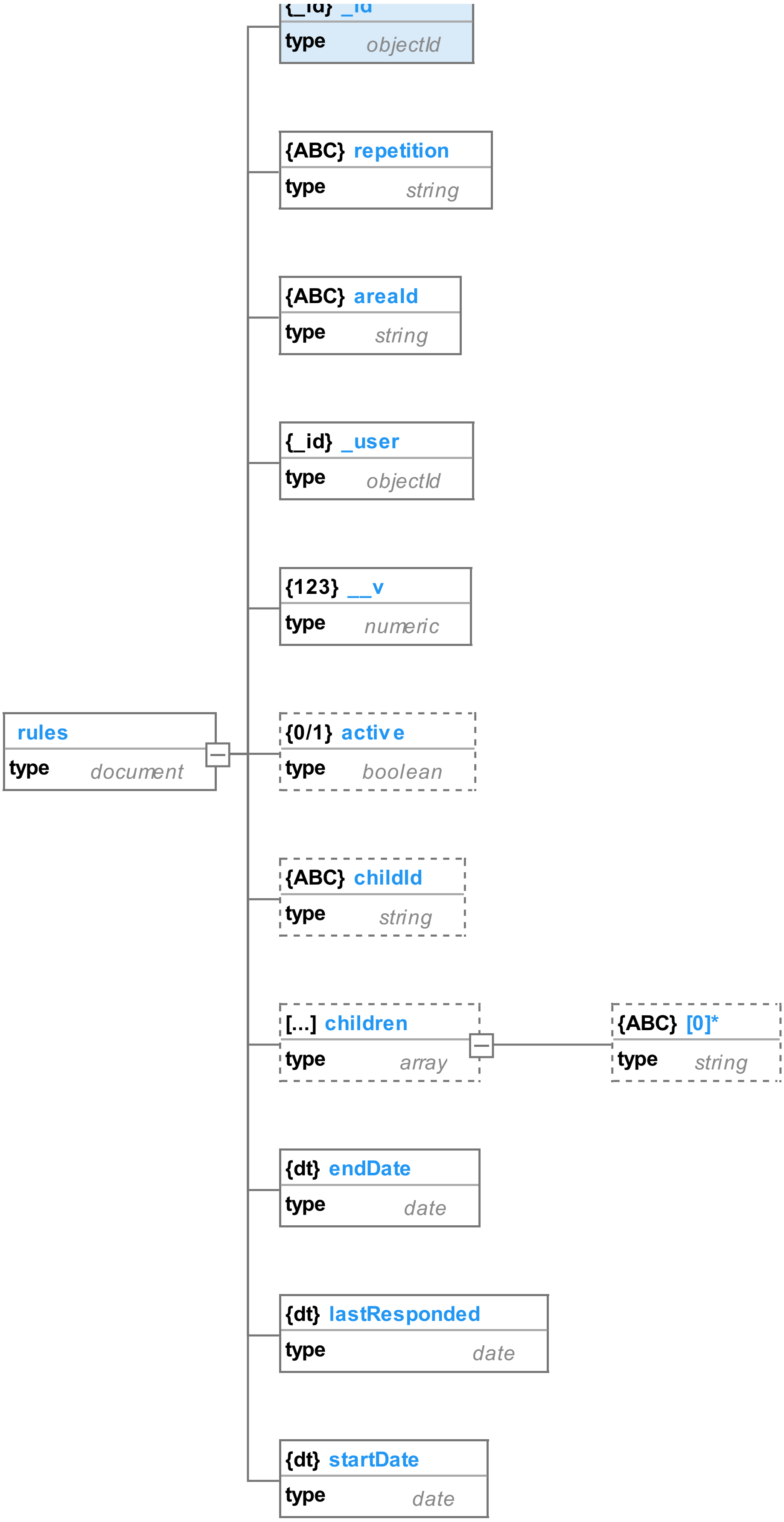
- \api\current_user - current signed in user,
- \api\logout - logging out user,
- \api\googleAuth - signing user in with Google OAuth,
- \api\register - registering new user,
- \api\login - signing user in,
- \api\rules - rules for current user,
- \api\rules\create - creating new rule for current user,
- \api\rules\read - rule data,
- \api\rules\update - updating rule,
- \api\rules\delete - deleting rule,
- \api\areas - areas for current user,
- \api\areas\create - creating new area,
- \api\areas\read - area data,
- \api\areas\update - updating area,
- \api\areas\delete - deleting area,
- \api\children - children for current user,
- \api\children\create - creating new child,
- \api\children\read - child data,
- \api\children\update - updating child data,
- \api\children\delete - deleting child data.

6 Data Base

6.1 Diagrams



Rysunek 47: Users document schema



Rysunek 48: Rules document schema

6.2 Queries for Rules

We will show implementation of queries only for rules, because queries of children and areas are implemented in a very analogous way. Rules queries are executed by mongoose library, therefore they are very simple.

6.2.1 Create

```
newRule = new Rule({
        startDate,
        endDate,
        repetition,
        areaId,
        lastResponded,
        childId,
        _user
});
await newRule.save();
```

6.2.2 Read current

```
Rule.find({
        _user,
        startDate: {$lte: Date.now()},
        endDate: {$gte: Date.now()}
});
```

6.2.3 Read all

```
Rule.find({
        _user
});
```

6.2.4 Update

```
Rule.updateOne({_id: ruleId},{
        startDate,
        endDate,
        repetition,
        areaId,
        lastResponded: Date.now(),
        childId,
});
```

6.2.5 Delete

```
Rule.deleteOne({
        _id
});
```