Article

Towards an Accessible Platform for Multimodal Extended Reality Smart Environments

Emanuela Bran [1,2,*], Gheorghe Nadoleanu [3] and Dorin-Mircea Popovici [4]

1 Faculty of Electrical Engineering and Computer Science, "Transilvania" University, 500024 Brasov, Romania
2 Black Sea Institute for Development and Security Studies, "Ovidius" University, 900527 Constanta, Romania
3 Faculty of Sociology and Social Work, University of Bucharest, 010181 Bucharest, Romania
4 Department of Mathematics and Computer Science, "Ovidius" University, 900527 Constanta, Romania
* Correspondence: bran.emanuela2019@gmail.com or emanuela.bran@unitbv.ro or emanuela.bran@365.univ-ovidius.ro; Tel.: +40-724-021-149

Abstract: This article presents the DEMOS prototype platform for creating and exploring multimodal extended-reality smart environments. Modular distributed event-driven applications are created with the help of visual codeless design tools for configuring and linking processing nodes in an oriented dataflow graph. We tested the conceptual logical templates by building two applications that tackle driver arousal state for safety and enhanced museum experiences for cultural purposes, and later by evaluating programmer and nonprogrammer students' ability to use the design logic. The applications involve formula-based and decision-based processing of data coming from smart sensors, web services, and libraries. Interaction patterns within the distributed event-driven applications use elements of mixed reality and the Internet of Things, creating an intelligent environment based on near-field communication-triggering points. We discuss the platform as a solution to bridging the digital divide, analyzing novel technologies that support the development of a sustainable digital ecosystem.

Keywords: technology democratization; application development platform; extended reality; intelligent environment; mixed reality; Internet of Things; smart sensors; multimodality; sustainable development; flow-based programming

check for updates

Citation: Bran, E.; Nadoleanu, G.; Popovici, D.-M. Towards an Accessible Platform for Multimodal Extended Reality Smart Environments. *Information* **2022**, *13*, 439. https://doi.org/10.3390/info13090439

Academic Editor: Gabriel Luque

Received: 22 July 2022
Accepted: 15 September 2022
Published: 18 September 2022

1. Introduction

The internet is constantly developing, and we are at the dawn of a new phase of dynamics between the participatory elements of the web and the structural evolution shaped by the driving forces promoting conscious change [1], leading to novel ways individuals and society engage with technology [2] and are impacted by digital transformation. This paper tackles the emerging digital context of connected smart things and hybrid-reality elements, proposing the DEMOS (Distributed Event-driven Modular Open Source; from the Greek word δήμος—*demos*, meaning common people of a democracy) platform as an accessible inclusive ecosystem for bridging the digital divide along different dimensions [3]. Throughout the article, we will focus on how the modularity of components offers new ways of application building and interaction with the digital sphere in the context of platform-based technology democratization and decentralization of collaboration. We will also take into consideration how distributed applications create new relationship dynamics between stakeholders.

This article comes as a fulfillment of previous research done in several areas of computer science and digital humanities, starting from conceptualizing levels and types of hybrid physical–virtual interaction within an open affordable mixed-reality (MR) platform [4]. Our previous research continued with envisioning a digitally transformed ubiquitous museum for sustainable development [5] and developing a distributed application for driving in the intelligent environment [6]. Within the subject of digital humanities, we previously tackled the digital transformation of Society 5.0 based on technologies of the 4th Industrial Revolution, analyzing micro and macro challenges and solutions [7].

This article is structured as follows. In Section 2, we discuss some of the existing automation and extended-reality (XR) platforms and key technologies that help create and sustain a digital transformation, paving the road to further explain our concepts and solutions in the following sections. Section 3 continues by addressing problems raised in the previous section, developing ideas, and discussing solutions based on existing or emerging technologies, focusing on the particular requirements of the implemented platform solution. In Section 4 we present the implementation and functionality of the DEMOS prototype platform together with the conceptual framework that builds on the previously discussed features and works out several improvements that leverage resources offered by smart things and services while mitigating issues of no-code to low-code application creation.

At the time this article was written, the platform's application-creation interface was in the design phase, and fully functional applications were created directly from compiled JSON configuration files that respect the core concepts and patterns explained in Section 4. We tested the core concepts and patterns of the DEMOS prototype platform by building two applications from JSON, the ConfiDrive App and the MusExp App presented in Section 5, and letting a sample of programmers and naïve students see the configuration and play with and rate the applications. In Section 6 the testing methodology involved the same sample of programmer and non-programmer students who were asked to design other demonstrative applications as part of an experiment to assess the affordance of the concept. In the second part of Section 6 students rated engaging platform features for application creators and explorers by answering a survey, opening the way for conclusions and future work directions.

2. Related Technologies and Conceptual Frameworks

Internet social platforms offer the possibility for people to engage with the digital world in multiple ways, blurring the lines between software developers, content creators, and consumers. Moreover, novel digital developments bring focus on the shift in relationship dynamics between the provider of services and the consumer client, creating the possibility of designing new business logic based on the platform economy [8] and governance solutions relying on trustless technologies [9].

In Section 2.1 we will discuss existing technologies and platforms that foster the creation of a smart environment and continue in Section 2.2 to analyze technologies that leverage the creation of decentralized platforms focusing on innovations that sustain digital ecosystems.

2.1. Automation Applet Creation Platforms

Institutions and organizations are currently striving to create a global intelligent environment that brings together smart things, artificial intelligence (AI), and XR technologies in order to better manage the planet's resources, create new spaces of interaction between people, and lead to a more conscious society [10]. Although the endeavor to connect things, people, information, and processes across the internet is a fairly achievable task for a team of developers creating an application, platforms that would offer this possibility for the general public that wishes to implement an innovative idea are still in their inception.

The Internet of Things (IoT) connects devices that are embedded in smart homes, smart cities, smart industries, and other intelligent environments. Producers targeting the smart-home industry offer smartphone interfaces such as the Ikea Smart Home App (IKEA Home smart app and TRÅDFRI gateway support – IKEA. Available online: https://www.ikea.com/us/en/customer-service/product-support/app-gateway/, (accessed on 23 June 2022)) and the Xiaomi Home App (MIJIA. Available online: https://home.mi.com/index.html, (accessed on 23 June 2022)) that provide tools for designing basic scene automation by setting an array of event triggers and the desired characteristics of the controlled IoT devices [11]. Automation platforms such as the Apple Home App for Homekit (Home app –

Apple. Available online: https://www.apple.com/ios/home/, (accessed on 23 June 2022)), Google Home App (Smart home automation from Google | Google Home. Available online: https://home.google.com/welcome/, (accessed on 23 June 2022)), and Amazon Alexa App (Alexa Smart Home - Learn about Home Automation | Amazon.com. Available online: https://www.amazon.com/alexa-smart-home/, (accessed on 23 June 2022)) offer naïve users an extended natural interface for designing multimodal smart interaction by connecting the producer's devices or other compatible devices and in-house or third-party intelligent services.

XR and AI augment the elements of the IoT by offering enhanced interaction [12]. Virtual reality (VR) enables the creation of immersive areas, and augmented reality (AR) provides a seamless natural interface between the user and smart things for multimodal interaction and visualization while quietly sensing their presence throughout the environment. Platforms such as SparkAR (Spark AR Studio - Create Augmented Reality Experiences | Spark AR Studio. Available online: https://sparkar.facebook.com/ar-studio/, (accessed on 23 June 2022)) offer tools for the creation of simple AR Instagram (Instagram. Available online: https://www.instagram.com/, (accessed on 23 June 2022)) filters with other more complex visual-programming platforms such as Vuforia (Vuforia Developer Portal |. Available online: https://developer.vuforia.com/, (accessed on 23 June 2022)) and Unity3D (Unity Real-Time Development Platform | 3D, 2D VR & AR Engine. Available online: https://unity.com/, (accessed on 23 June 2022)), targeting the full XR spectrum [13].

AI, also a key element in application development, is often involved in human–machine interaction for reading human cues and offering human-like information-processing capabilities. Examples of code-free flow-based visual-programming platforms for AI are Levity (Levity | No-code AI workflow automation platform. Available online: https://levity.ai/, (accessed on 23 June 2022)) and Amazon Machine Learning (Machine Learning and Artificial Intelligence - Amazon Web Services. Available online: https://aws.amazon.com/machine-learning/, (accessed on 23 June 2022)) [14]. AI-processing features are also available in the engines of AR platforms for image and scene recognition, and on general-purpose automation platforms for machine-learning (ML) tasks.

A state-of-the-art automation platform that combines the previously mentioned features is Node-Red (Node-RED. Available online: https://nodered.org/, (accessed on 23 June 2022)), which offers no-code to low-code flow-based programming, with the possibility of adding new nodes to the platform [15]. It is based on Node.js (Node.js. Available online: https://nodejs.org/en/, (accessed on 23 June 2022)) and is used both in automation for the industrial IoT and in innovative applications that enhance the workplace or home environment with intelligence. Other more limited platforms such as Zapier follow the "If this then that" (IFTTT) (IFTTT. Available online: https://ifttt.com/, (accessed on 23 June 2022)) approach and provide means of designing applications for logistic management and simple automation tasks [16].

2.2. Digital Technologies for Platform Ecosystems

The World Wide Web started as static webpages and developed into web-based services that usually offer dynamic user-tailored free-of-charge content. This is achieved by the big technology corporations through storing personal data that are further used for marketing purposes that cover the cost of the provided services [17]. Even though such services are asking for permission to store and process data, with partial transparency regarding the purpose, society is raising concerns about the centralized governance of web services.

Innovation is carried out by creating a more decentralized web, which may involve some form of blockchains [18] or peer-to-peer network systems [19]. The semantic web was a concept regarding the future evolution of the web towards machine-comprehensible content, but current distributed-ledger technologies already make possible machine-operable web activities in terms of secure authentication, cooperation, and value transaction. The Ethereum network (Home | ethereum.org. Available online: https://ethereum.org/en/, (accessed on 23 June 2022)) and others that followed provide smart contracts for process automation and special tokens to represent the value of digital resources.

The non-fungible-token economy [20] seems to be on the rise due to the blockchain and metaverse hype, but the technology is nevertheless a future valid form of ownership proving for digital assets. Crypto-coins (Cryptocurrency Market | Coin Prices & Market Cap | Binance. Available online: https://www.binance.com/en/markets, (accessed on 23 June 2022)) developed under different strategies such as the inflationary or the deflationary model [21], together with the innovation of consensus mechanisms that try to solve technical issues related to network distribution while mitigating the impact on climate change and power concentration, are bringing value to the network and are creating opportunities for decentralized application ecosystems to thrive.

We have seen how simple platform-based services such as Uber (Earn Money by Driving or Get a Ride Now | Uber Romania. Available online: https://www.uber.com/ro/en/, (accessed on 23 June 2022)) and Airbnb (Vacation Homes & Condo Rentals - Airbnb – Airbnb. Available online: https://www.airbnb.com/, (accessed on 23 June 2022)) disrupted the classical offline business models [22], and decentralized finance will eventually seize the present opportunities and create a sustainable digital transformation on many levels of society. Accessible platforms that are promoting the democratization of digital technology [23] are necessary for inclusive social change along with a conscious ethos of participation in the new hybrid reality [24].

3. Context and Considerations

Digital innovation drives global change from the micro to the macro level of society. The United Nations 2030 Sustainable Development Goals (SDG) [25] highlight the importance of digitalization for the welfare of people in general and for creating a global network of cooperation between governments, civil society, and the business sector. The World Economic Forum devises an annual Global Risk Horizon report [26], which includes digital technology-related threats that are strongly linked to other vulnerabilities. The European Union [27] and other international organizations analyze the impact of digital technology, keep track of digital developments, and coordinate planned action toward the envisioned future of society.

The digital technological gap spans regions and countries, and leaves outside the global society vulnerable groups of people. It implies several dimensions, such as access to technology, technological literacy, and the power of governance over technology. In high-gross-domestic-product (GDP) countries, risks involving technologies concern cybersecurity issues, infrastructure breakdown, and adverse technology outcomes [28], whereas in lower-GDP countries risks involve the digital inequality gap [29]. There is a growing need for inclusive digital technologies and opposition toward centralized, monolithic governance over digital products. Issues of access and power related to technology can be solved by creating solutions that pave the ground for a sustainable digital transformation.

Regional development and empowering vulnerable groups of people are targets achievable through creating new digital platforms and economic models. Inclusive platforms shall offer an array of roles to perform services comprising an ecosystem of viable prolific change on several levels of society. Development may target improved education, job creation, equality of employment, environmental protection, active participation in democracy, and a growing sense of being represented in the global landscape (see Figure 1). Transparency and interoperability between different governmental, civil, and business stakeholders are achievable by creating interconnected interfaces over a digital platform, which represents target number 17 of the SDGs.

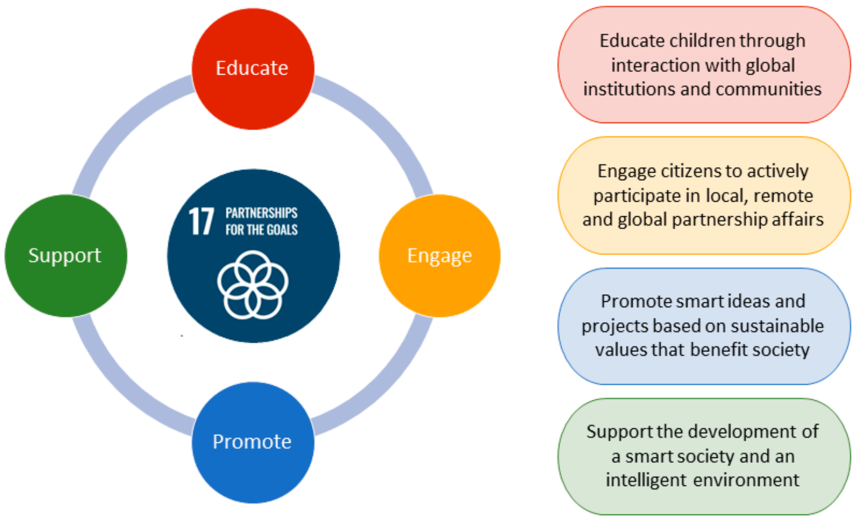

Educate
Engage
Promote
Support
17 PARTNERSHIPS FOR THE GOALS
Educate children through interaction with global institutions and communities
Engage citizens to actively participate in local, remote and global partnership affairs
Promote smart ideas and projects based on sustainable values that benefit society
Support the development of a smart society and an intelligent environment

Figure 1. The role of a digital platform in achieving the SDGs by offering an environment for educating, engaging, promoting, and supporting partnerships between stakeholders.

In this section, we will cover how to achieve a sustainable digital transformation by harnessing the latent potential within society (see Section 3.1) and the characteristics of a digital platform for enhancing the life experience in contact with the intelligent environment (see Section 3.2).

3.1. Harnessing Social Resources for a Profound Digital Transformation

Technology evolves from experimental narrow field applications to worldwide mainstream everyday use. XR-spectrum technologies [30] first appeared in our imagination in 1901, and in the '50s and '60s materialized in the Sensorama VR machine [31]. Afterward, AR technology emerged, which is slightly more complex in that the virtual elements need to be anchored onto physical reality. It was first designed by researchers and intended for industrial use, and later as it gained momentum, it opened to the public due to being marketed by the tech-industry giants.

Several waves of development followed, including mobile AR software, mobile AR hardware, tethering smart glasses, and standalone smart glasses. By consulting the Gartner Hype Cycle [32], we may note the hype phase of AR between 2010 and 2011, before the technology followed a downward trend. This curve characterizes most cutting-edge technologies and is mostly related to the commercial excitement produced by the media promotion conducted by big tech companies investing in development [33], which hits the contextual limit of society's capacity for technological adoption for that specific version at that moment in time.

Mark Weiser said with respect to ubiquitous computing and in contrast to the virtualization paradigm [34] that the most profound technologies are those that disappear, weaving themselves into the fabric of everyday life until they become indistinguishable from it. Facebook was recently renamed Meta and is struggling to capitalize on the social aspects of XR both in terms of app usage and content creation, marketing the metaverse as a virtually extended space of experience connected to the physical reality through digital twins. To be able to access the potential of XR market adoption, technology should become accessible to the public as consumers and creators, and be included within the culture and worldview of people by offering multiple intuitively customizable features.

No-code or low-code collaborative platforms are key in achieving the leap towards wide adoption [35]. Furthermore, creating distinct roles for the stakeholders and a strategy of financial incentives secures the development of the digital ecosystem [36]. Solutions often require the modularity of components and may involve distributed ledger technologies [37]. With the emerging IoT augmented by blockchain technologies to become the Internet of

Trusted Things [38], modularity and decentralization are starting to build on the power of edge computing and become a potential alternative to centralized system architectures.

We should point out that centralization and decentralization are nuances that may be applied on many levels of an organization and involve a continuum comprising hybrid solutions [39]. There are flavors of decentralization that target hardware architecture, software architecture, or social systems behind the driving forces that govern a network [40]. Accessible platforms [41] that engage high-profile government figures, business stakeholders, and the general public representing civil society bring decentralization and democratization to all levels of the digital system.

One issue uniting humanity is the problem of climate change, a top priority for many international organizations, and we may add that the problem of pollution, healthy food, and even access to clean water is a globally undeniable issue [42]. A digital platform that engages children, experts, scholars, activists, and politicians may bring a sustainable global change by creating a circular economy [43] starting at the deeper level of people's mindset [44]. Platforms may provide a space for educating people, promoting awareness, creating cultural bridges, offering new modes of cooperation, and incentivizing good action [45].

Blockchains based on social and environmentally conscious consensus mechanisms such as proof-of-stake and developments beyond [46] are able to sustain the emergence of a platform-based economy of services [47]. This represents an opportunity [48] for developing countries to move towards a cleaner economic activity, and for developed countries to create remote job opportunities that promote healthy lifestyles near nature. Overall, the democratization of digital technology decentralizes society and promotes sustainable development in remote areas.

In order to generate a sustainable societal impact, a digital platform should aim to bring to life the potential of the vast internet resources by creating endpoints for all four internet pillars: people, things, processes, and data. Social networks, smart things, logistics systems, and data visualizations may all be part of our daily lives for hands-on interaction within an accessible digital platform, conceived in a modular fashion with flow-based visual programming that enables ideas, necessities, and desires to turn into sharable, reconfigurable, monetizable applets.

3.2. Characteristics of a Platform for a Digitally Enhanced Life Experience

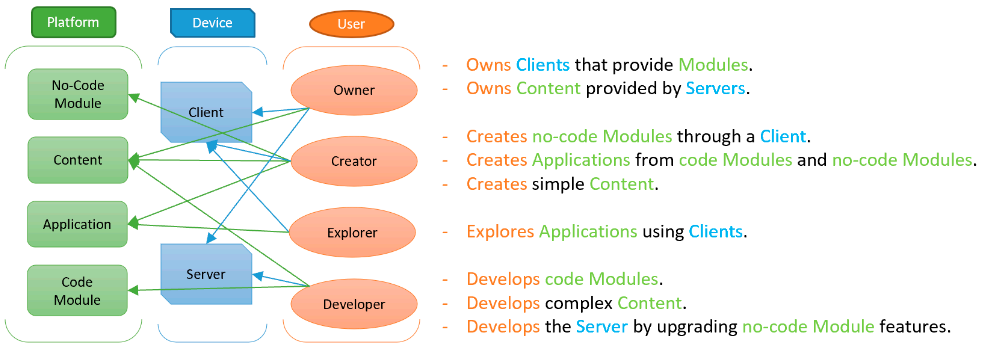
Applications will be available after creation on the platform under several types of sharing options that target ways of discovery, multiuser handling, and payment method. The platform requires features of social networks so that applications may be promoted by other users sharing them in addition to using standard name-queries. In addition, applications may be intended for single-user purposes or multi-user scenarios and be either free of use charge or need some form of payment. Again, the modularity of the system will make these configurations both necessary and possible, and we will focus on how modularity imposes special traits in discovery, concurrency, and payment.

When discussing enhanced application discovery [49], an important search filter is showing relevant applications by the devices enlisted in their configuration. We should note that the user does not need to own all the devices enlisted, since due to the modularity of the system, the application will function in a distributed manner across the network. The user may thus either own a device that creates information input in order to control some other device's output, may own a device that visualizes some output from the networked input services and devices, or may support an application to function by sharing their hardware and hosting necessary intermediary components of the application. Applications may be intended for multiple user interactions [50] at one time, requiring diverse strategies for managing application availability depending on the configuration of the interaction.

Ways of discovering the applications should also take into account circumstantial information regarding the user [51]. In an intelligent environment, services follow the user seamlessly across different places while adapting to the new circumstances. We consider this platform to be offering microservices that should become available, discoverable, and highlighted to the user at the right moment, represented by fuzzy or crisp logic-set operators [52] of the complement, intersection, and union over several dimensions such as location, time, and interactions within the platform [53]. Applications are either available or potentially available with the help of a user joining the services and hardware already set in place. Both distributed apps and standalone apps, depending on their purpose and configuration, may function across the network or be restricted to a location.

Practical means of binding applications to locations are near-field communication (NFC) tags, quick-response (QR) codes, and geolocation boundaries [54]. By scanning a code or letting geolocation notifications pop up, the user may be guided to connect to a standalone application that functions on their smartphone, or the user may be directed to interact with a distributed application that uses special hardware installed in that location. Applications may also have all necessary hardware already in place, and scanning would then only be aiding the application with user data sharing by signing the transaction using the private key for data available on a distributed ledger. Standalone applications will be able to function simultaneously on multiple devices without any conflicts, whereas distributed applications that require the user's devices to connect for interaction need special management for concurrency.

As seen in Figure 2, users may take on several roles [55] within the platform that may include providing or consuming resources. Due to the modularity involved, *creators* may be different from *owners* and *explorers* may become creators just by interacting with the platform. Resources that are involved with the platform comprise the hardware that helps application components run, including the servers that host resources and aid communication. Software resources developed by programmers include input components that connect to sensors and services and multimodal output components. An accessible platform offers the possibility to design processing nodes with no-code visual programming by naive users without programming experience, and *developers* may also add special-purpose processing nodes to the platform. Other resources may include content such as multimedia and information regarding users' activity and data in general, such as IoT generated or from other sources, which may also aid ML.



- Owns Clients that provide Modules.
- Owns Content provided by Servers.
- Creates no-code Modules through a Client.
- Creates Applications from code Modules and no-code Modules.
- Creates simple Content.
- Explores Applications using Clients.
- Develops code Modules.
- Develops complex Content.
- Develops the Server by upgrading no-code Module features.

Figure 2. The roles of a user in relationship with the platform's components and the devices that contribute to the distributed ecosystem.

Remuneration of application creation, content uploading, hardware sharing, and even interaction within the platform and user data publishing promote sustainable growth of the platform ecosystem. Blockchain technology helps automate payments using smart contracts, and accounts represented by public keys may belong to either direct human users or smart things owned by organizations. Automated micropayments [56] with quotas dictated by algorithms that target the community's thriving may engage explorers and creators alike for spending and earning tokens. Explorers and creators should have the possibility to overwrite algorithm-set quotas and replace automatic charging by prompting the smart contract to the other party and waiting for agreement. Reversing payments between explorers and creators would be useful in the testing phase of an application or specially designed campaign events where explorers earn cryptocurrency by participating and collaborating inside the intelligent environment.

4. The DEMOS Prototype Platform as Proposed Solution

We chose to work with a Node.js server in order to generate frontend JavaScript (ECMA-262 - Ecma International. Available online: https://www.ecma-international.org/publications-and-standards/standards/ecma-262/, (accessed on 23 June 2022)) code according to JSON (ECMA-404 - Ecma International. Available online: https://www.ecma-international.org/publications-and-standards/standards/ecma-404/, (accessed on 23 June 2022)) file configurations. Core modules that were used are the Express.js (Express - Node.js web application framework. Available online: https://expressjs.com/, (accessed on 23 June 2022)) server that handles the back-end tasks and Socket.io (Socket.IO. Available online: https://socket.io/, (accessed on 23 June 2022)) for communication between devices within an applet. The applets run inside the browser, and depending on the services that they require, may or may not need an internet connection. We believe that by implementing two distinct applications we demonstrated the proof-of-concept for a collaborative low-code/no-code platform for creating accessible multimodal XR applets for smart devices.

The platform provides a collection of input and output nodes, templates for configuring different types of processing nodes, and a module for communicating between devices in order to achieve the functionality of distributed applets. Our main focus in the implementation of the prototype platform was to select the most easily understandable visual-programming patterns and the most useful templates for building modular applications based on information flow. As for the application-creation interface, we conceptualized ways of exposing the patterns to the user and providing the tools for configuring the nodes. The server interprets JSON configurations of processing nodes and node links and correctly generates JavaScript code for each device as a demonstration of functionality for the prototype platform and the conceptual framework.

This section will present the components targeting the user experience as a creator and explorer of the smart environment, the technical components that help achieve intra-device and inter-device communication together with the compiler that concatenates the communication module with the modules generated from JSON configuration files. We will explain the concepts offered for non-programmers to design nodes that are decision- and formula-based information processors and the two demonstrative applications' specific functionality by indicating the modular interconnected components.

4.1. User Experience as Creator and Explorer

Figure 3a presents the main interactions between the elements of the platform in a scenario wherein two devices are connected for different purposes, as we will describe in the current section. The platform has three main functionalities: to provide the users with an interface for application creation (see Section 4.1.1), to provide the users with an interface for exploring the applications (see Section 4.1.2), and to provide the devices with and means for inter-device communication (see Section 4.2.1) and the necessary configuration information (see Section 4.2.2). Every application is first created, and then it is available for exploration. While the users are exploring the application, the server and devices operate together in order to accomplish the tasks indicated by the JSON configuration file that helped generate the code in the first place.

The Express server provides devices with the specific resources that the client requires through the GET and POST methods. The resources are involved in the *App creation interface* (see Section 4.1.1) and *App explorer interface* (see Section 4.1.2). Each device that is part of a distributed application has a *Communicator module* (see Section 4.2.1) linked to the Socket.io server and specific *Application modules* (see Section 4.2.2) provided by the Express server. The most important component of the Express server is the *App builder compiler* (see Section 4.2.2), which generates and assembles the application components for each client device and sends them as resources.

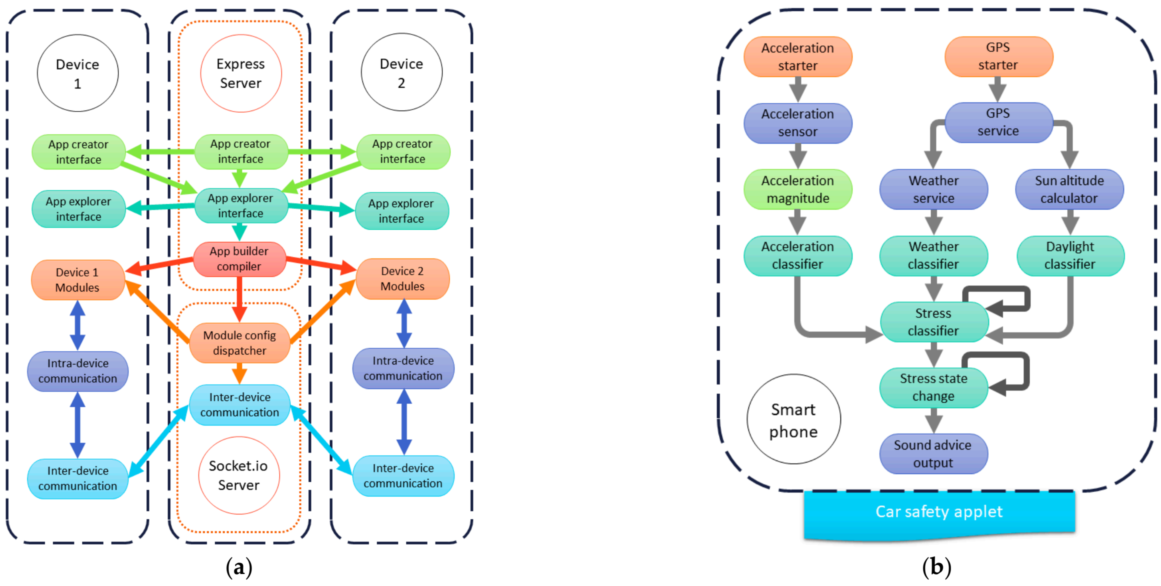


Figure 3. (**a**) The platform's core components and their interaction when two devices are connected. (**b**) Oriented graph of the data flow for a standalone mobile application for safe car driving (ConfiDrive) that was created and explored using the DEMOS platform.

4.1.1. Application Creation Interface

Elements of this platform may be split into fully configurable no-code elements by the applet creators and low-code elements that require programming skills. The information input nodes require a programmer to write code for fetching service data and capturing sensor data, and intermediary nodes also may use special libraries for complex calculations. The information-output nodes require code for visualizations and multimodal sensory output of digital assets, actions that are handled by libraries. Nonetheless, these nodes also offer codeless configurable features such as the volatility of information.

Figure 3b shows the visual scheme of the ConfiDrive applet that was built with the DEMOS platform. The application uses acceleration-sensor data and weather geolocation-based data in order to infer the level of stress for the driver and produce notifications for relaxing or countering drowsiness. Information flows along the arrow lines from top to bottom and the nodes that require programming skills are colored in light indigo blue. Three of the nodes are positioned at the beginning and the end of the array of information processing, with two of the nodes residing between the extremities of the node array. This shows that although the beginning and the end of an application will most likely need specially programmed nodes to harvest and visualize information, other codeless-processing nodes designed by naïve users may be interleaved within the information flow.

For the present version of the platform, we designed nodes to have only one type of output, which is labeled using the node's name, and a value that may be a primitive data type, such as a number or a string, or a structured object with multiple data fields. When the output of a node is an object, the user will select the specific field that is used as an input to another node, together with other fields of the same node or other input nodes. The output of a node may also become an input to itself in more complex configurations that we will discuss later in this section.

The power of the platform relies on the capability of providing accessible ways for non-programmers to connect the specially programmed nodes with totally configurable codeless nodes that they construct using the *App creation interface*. When a user wishes to create an application, they are presented with a visual interface. Behind the multiple ways of configuring nodes, there are two main JSON configurations, for numerical (formula-based) and nominal (decision-based) processing, with a third one (the general node) that we later devised while merging the first two configurations, which is also the most powerful way of constructing nodes.

The first configuration presented in Figure 4a handles mapping the input data to a specific output by using a formula that may be either queried or inputted using an equation builder. The user may select data from a single input node or merge data from several input nodes. In this case, the components of the input are the x-, y-, and z-vector magnitudes of the acceleration, and the result of the formula is a single value of numerical type. We may note that another node, the one calculating the altitude of the sun, has an output in the form of an object. Its input consists of GPS coordinates represented by two numerical values, the latitude and the longitude of the car, whereas the output is also comprised of two numerical values, the altitude and the azimuth of the sun.

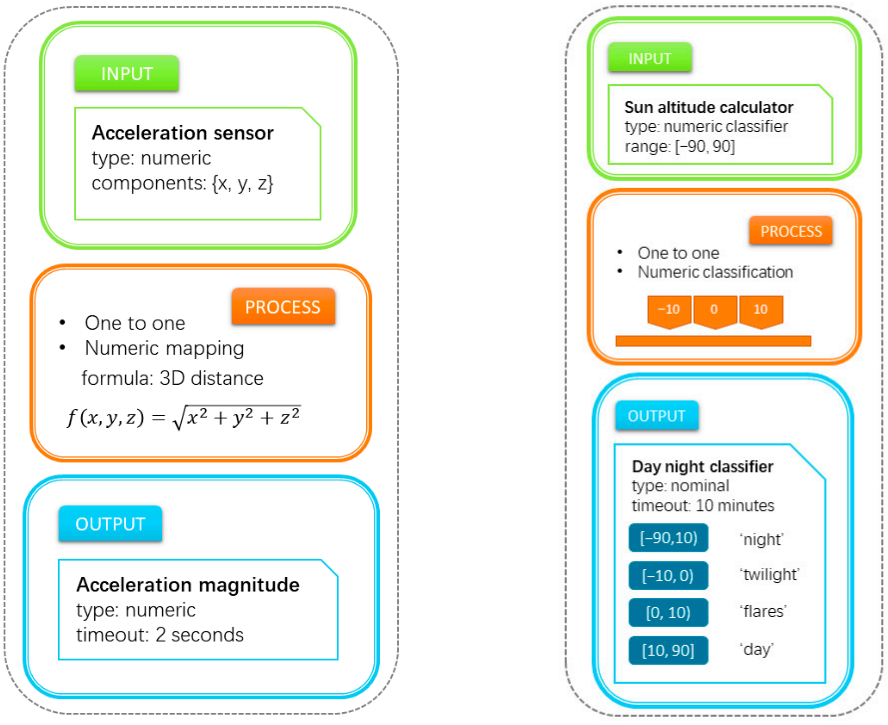


Figure 4. Numeric input-processing types. (**a**) Configurable node with a processing function based on the 3D distance formula with numeric output. (**b**) Configurable node with decisional processing based on a classifier with nominal output.

The second configuration presented in Figure 4b also handles numerical inputs, but the output is in the form of nominal values that represent categories of daylight. The sun's angle of altitude to the ground helps us classify the natural lighting, which may be daylight, sunset and sunrise flares, twilight, or night. Since the input is a single numeric value, the user selects points on a scale that will generate different output categories, which will be filled in with the specific string data-type value. The user may also manually enter the intervals or specific values for each category.

A complex variant of the second decisional-processing configuration is shown in Figure 5, where the user specifies three nodes as input along with the output of the present node computed inside a feedback loop as the fourth input value. Thus, stress is computed using the acceleration, weather, daylight, and stress values. The user then selects the number of output cases, which in this example is also four, and begins to create logical expressions related to the input values. An example is if the node has been in a high-stress state for more than 5 min already, the acceleration is low, and the weather is good, then the node will move into the normal stress category with a volatility of 24 h unless other events are received by the node.

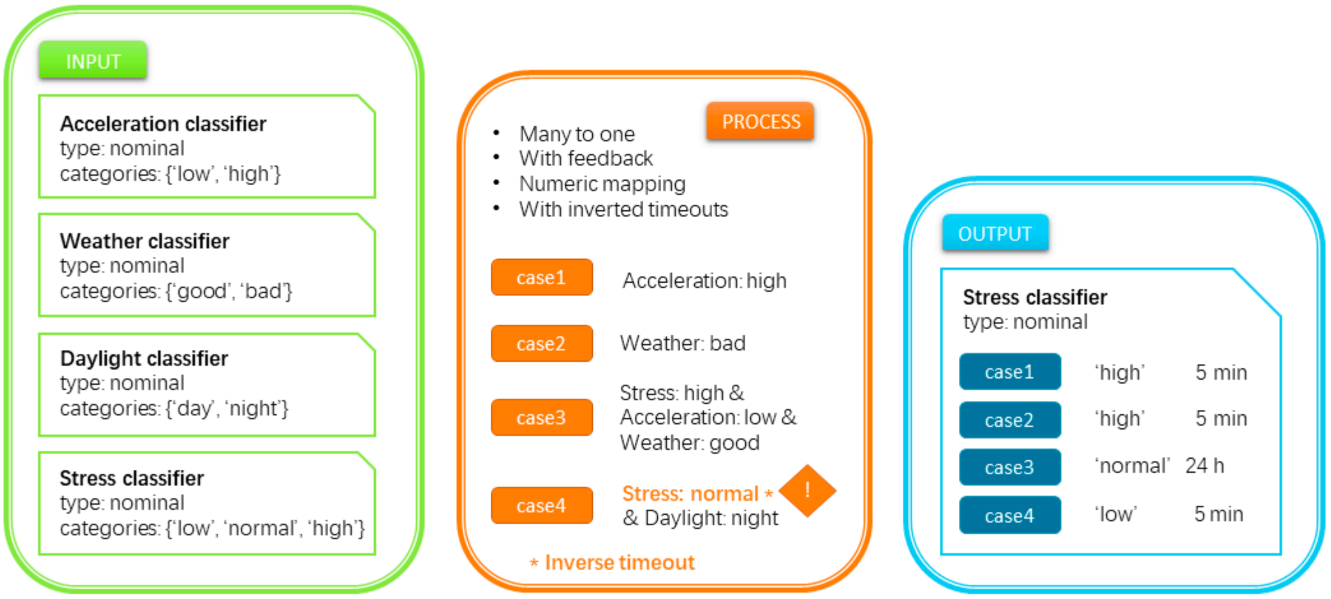


Figure 5. A complex node that processes nominal data and functions as a timed-state machine by merging input data from three other nodes with its previous output and taking into consideration time-related information, including inverting the timeout period.

One more feature is related to timeouts, which may be signaled to be computed in the regular timewise manner or the inverted manner with the help of a Boolean flag and the XOR operator. The regular manner is used in the first case, where if acceleration is high the node goes into high stress for 5 min, which may be extended every time a new high-acceleration event is produced until there is no such event detected for 5 min so that the high-stress state will timeout and enter the default normal stress state. The reverse timeout is the one computed in the fourth case, where after the node has been in the normal stress period for 5 min and if it is night, only then will the node enter the low-stress state, which is dangerous due to driver drowsiness behind the wheel. Thus, only after the normal state has reached timeout may the state change, opposite to events that are valid only before reaching timeout.

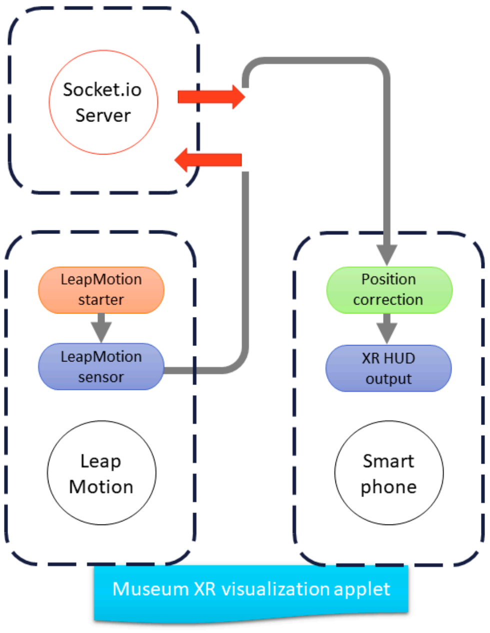
Once the user enters the platform as a creator, the interface provides several helpful resources and tools for application creation. The tools include the elements present in the graphic user interface for node configuration and the tools for linking nodes into an oriented graph within a distributed application that produces standalone applications and distributed applications. When linking nodes with a device or between devices, the user specifies the set of nodes that produce input for the currently selected node and the fields of the message the node will listen to, such as subfields within the object's value and time-related information. The user will also have connectivity patterns available, as we will continue to explain in Section 4.1.2, and node templates to choose from, as presented in Section 4.2.

4.1.2. Application-Exploration Interface

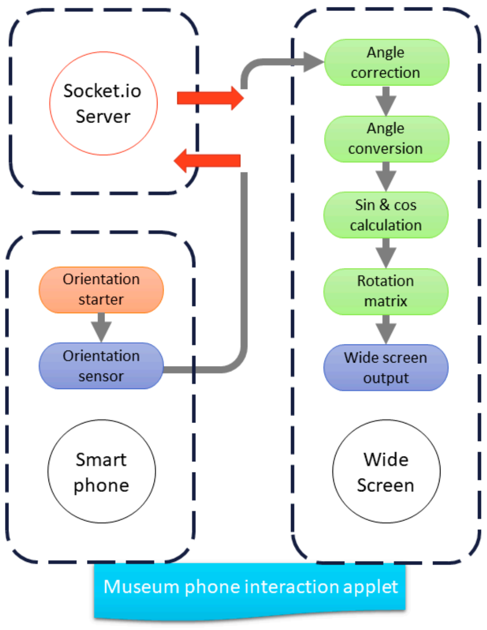
In this section, we will focus on the features that make the exploration of the applications a natural user experience. The central concept used in exploration is that applications are embedded within the environment. QR codes and NFC tags both contain information that may be used to anchor application features to specific applet-access places. For our demonstrative applets, we used NFC tags since scanning a tag requires less action from the user than reading a QR code. Conditions based on other more-complex parameters such as geolocation, time, and interaction may also emulate anchoring applets to specific contexts.

The ConfiDrive application, which is used for safely driving a car, may be enabled and disabled by scanning an NFC tag present in the car. An example of a set of more-complex conditions is using the car applet only when driving outside the city of residence at night, which requires creating a node with this set of conditions, which will either enable or disable the applet on a phone. Such a node will simply calculate whether the distance of the present geolocation to the city center exceeds a certain boundary and whether the sun is below 10 degrees of altitude. We may observe that the second condition would render useless the node calculating the altitude of the sun; thus, nodes may also be disabled from the explorer's side due to the modularity of the application.

We proceed to present the main components from the MusExp application, namely, the XR visualization part (see Figure 6a) of the MusExp application that illuminates a virtual artifact seen on an HUD reflecting the user's smartphone screen according to data coming from the LeapMotion device that detects the user's finger position, and the interaction part (see Figure 6b) of the MusExp application that rotates an artifact displayed on a wide screen according to data received from the user's smartphone orientation sensor. The museum tour consists of the two access points of multimodal interaction and smart points for the narrated output of information related to artifacts. These are applets designed to enhance the user experience of the visitor exploring the smart-museum environment and are a fit example to present some of the interactions detailed in Table 1 that a user performs in the role of an explorer.



(**a**)



(**b**)

Figure 6. Oriented graphs of data flow and the device part of the distributed MusExp application that was created and explored using the DEMOS platform. (**a**) The XR visualization part of the MusExp application that illuminates a virtual artifact seen on an HUD reflecting the user's smartphone screen according to data coming from the LeapMotion device that detects the user's finger position. (**b**) The interaction part of the MusExp application that rotates an artifact displayed on a wide screen according to data received from the user's smartphone orientation sensor.

Table 1. The interactions a user may perform while exploring the smart environment by scanning NFC tags, detailed by interaction participants, sequence of action together with purpose, and the working-logic sequence.

Interaction Participants	Action and Purpose	Logical Sequence
User's device, the app	A user will enable and disable their app, with each scan changing the app's state. - Good for standalone apps activated in a specific location and personal devices listening to external events	A node on the smartphone holds a Boolean value that enables and disables the starter nodes or the nodes listening for external events. The first time the app is enabled it requests the necessary code from the server.
User's device, the app	A user will enable and reenable their app, extending the active state while repeatedly scanning the tag placed in proximity. - Good for standalone apps and personal devices listening and producing events only while present in a specific place	After scanning the tag, a node on the smartphone holds a Boolean value with specific volatility, which automatically disables the node when reaching timeout.
User's device, the app	When scanning an NFC tag, a user will trigger an event within the app using multiple tags spread throughout the environment. - Good for tours, XR, and IoT apps that map the natural environment	Through the information present within the scanning tag, the event triggers an action such as visualizing particular content or moving a state machine to the next step (just as any other connected node works) by letting the tag information flow forward.
First user's device, one shared device, second user's device	The first user will connect to the shared device. When the second user connects to the device, they will disconnect the first user. - Good for apps that require a single user to control a device in a shared environment	When scanning the tag, the first user's device holds the timestamp and sends it to the shared device. All events coming from them have the timestamp information and pass the shared device's filter until the second user scans the tag, and the device will filter by the updated timestamp.
First user's device, one shared device, second user's device	The first user will connect to the device, then the second user will also connect, and on the device their inputs will be merged. - Good for apps that require the collaboration of multiple users for the same task	When the users scan the tag, they get a timestamp for identification, which is known by their device and the shared device as well. Their inputs are then merged with the help of a formula-based processing node on the shared device.
First user's device, one shared device, second user's device	The first user will connect to the device, then the second user will also connect, and each user will control a specific component. - Good for apps that require the collaboration of multiple users executing distinct tasks	The users will each scan different tags in order to enable distinct components of the distributed application and collaborate using distinct input or output components.

Within the ConfiDrive and the MusExp applets' context, we may enumerate several patterns presented in Table 1. The application for safely driving the car is enabled with the help of the first interaction pattern. The second pattern is used in the museum for visualizing the hologram of the artifact while the phone is placed in the HUD. The third pattern helps to trigger multimedia information on the smartphone as an enhanced tour experience. The connectivity pattern present in the fourth example from the table helps users connect one by one separately to the screen in order to control the virtual artifact with their phone. The fifth pattern may be useful in a smart-museum context where users would have to synchronize action in order to move a heavy virtual object such as a virtual gate. Finally, the last connectivity pattern is designed to bring together users serving diverse roles and interacting through various devices and collaborate within a single complex application, such as by connecting the two MusExp applets in Figure 6: One user rotates an object with their phone and the other illuminates the object with the help of the LeapMotion sensor in order to find a hidden symbol in a gamified museum experience.

4.2. *The Backend Functionality and Assembled Code*

Applications are created through the visual interface that produces JSON files, which are interpreted by the compiler delivering JavaScript code for each device that is part of the distributed application. The compiler present on the server-side concatenates several types of code: the module for intra-device communication, the special node modules that were written by programmers, and the processing node modules that were designed by nonprogrammers. The Express server that delivers compiled code and the Socket.io server that provides communication between the assembled components assure the backend functionality of the platform.

4.2.1. Intra and Inter-Device Communication

The main feature of applications created within the discussed platform is information flow from a set of nodes to their connections. The flow is triggered by starter nodes attached to the special nodes that read sensor data or request information from web services. The starter nodes are designed by the non-programmer, who specifies how often the starter node will trigger the sensor or service node, or in the case that no timer is specified, the node will use the default value recommended by the programmer building the sensor or service node.

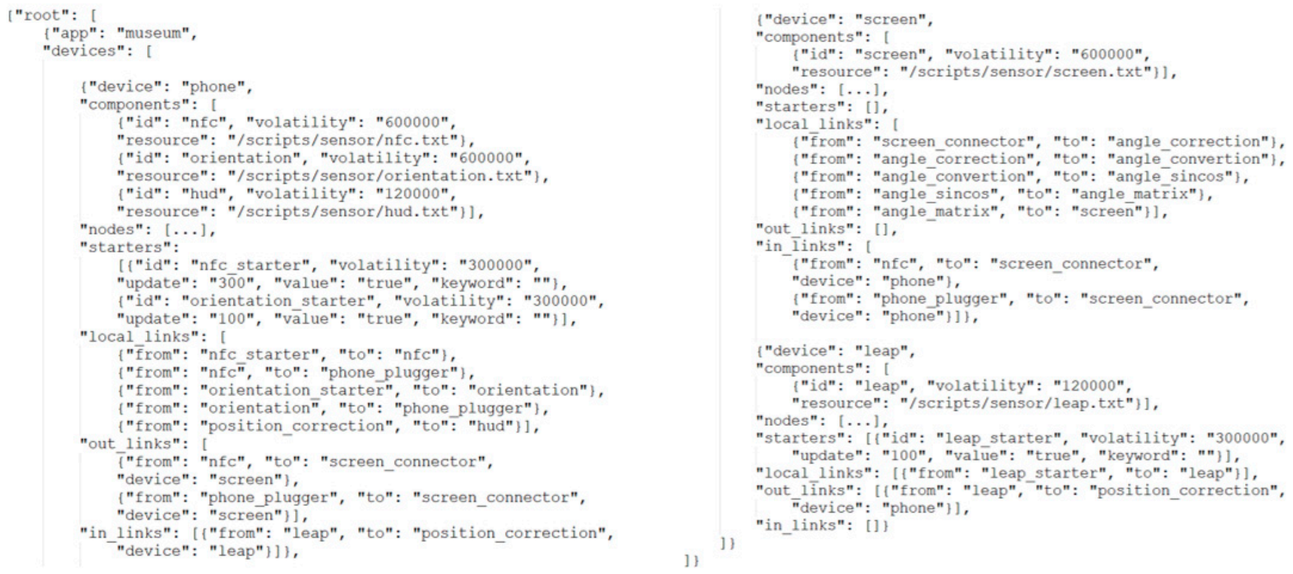
The non-programmer application creator user may also specify the volatility of the messages emitted by each node used in the application. Special nodes have a default value specified by the programmer that may be changed by the creator as desired, and processing nodes require the creator to specify the volatility of each output field. In the case of formula-based numeric processing, there is only one output field, whereas in the case of decision-based nominal processing, there are usually multiple output fields, each of them having the possibility of a different volatility value to be set and a Boolean flag to reverse the logic of timeouts.

Messages created by the nodes contain the name of the node that created them, the volatility of the event, and the value, which may be a primitive data type or a complex object with multiple data fields. Each processing node holds in its memory the data received from other nodes, organizing data by sender-node name. The nodes are designed to listen to specific subfields of data within the value field of the updated memory and execute a new computation each time a new event triggers the node as long as message timeouts still respect the time-wise scheme.

The devices within a distributed application are grouped in a single chatroom, and the Socket.io server emits received messages according to the configuration scheme. When a node emits an event, the message travels to the communicator intra-device module, where the sender–receiver couplet is checked against a list. If the pair is an intra-device link, then the communicator component triggers the internal function of the receiver-processing node module, which handles the processing of the information. If the pair is an inter-device link, the current device headers are added to the message, and then it is stringified (i.e., cast to the string-data type) and sent to the Socket.io server, where it is directed toward the destination device. When it is received by the communicator module of the destination device the message is parsed, and the internal processing function of the targeted node module is triggered.

As depicted in Figure 3a, the Express server assembles the code for each device part of the distributed application and delivers the JavaScript module nodes. When a device fully loads all modules, it dispatches an “initialization” event to the Socket.io server, which responds with the schema of initialization. The communicator component of the device then initializes the internal intra-device module links and the external inter-device module links. The intra-device links contain actual triggers for the internal functions of the specific modules initialized using the “eval()” JavaScript command. This is the only moment that this command is used in the entire information flow so that the applications run optimally.

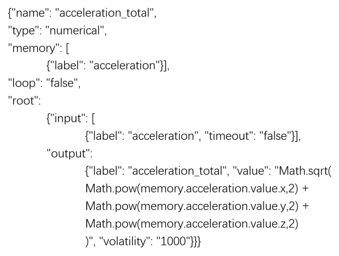
JSON files are used in inter-device communication as JavaScript objects and converted into strings while being dispatched by the Socket.io server. They are also used for storing the application configurations interpreted by the compiler in order to build the application, which is delivered by the Express server. In addition, they are used by the Socket.io server to initialize the communication schema on each device. Applications are stored in a JSON file in a "root" field containing an array, each element having an "app" field containing an identification name for the application followed by a "devices" field containing the array of configurations for each device. Each device configuration has a name, a list of code dependencies for the special nodes, and the actual configuration designed by the application creator, consisting of the oriented graph of links between nodes (listed in Figure 7) and the processing node characteristics, which will be presented in Section 4.2.2.



```
{"root": [
    {"app": "museum",
    "devices": [

        {"device": "phone",
        "components": [
            {"id": "nfc", "volatility": "600000",
            "resource": "/scripts/sensor/nfc.txt"},
            {"id": "orientation", "volatility": "600000",
            "resource": "/scripts/sensor/orientation.txt"},
            {"id": "hud", "volatility": "120000",
            "resource": "/scripts/sensor/hud.txt"}],
        "nodes": [...],
        "starters":
            [{"id": "nfc_starter", "volatility": "300000",
            "update": "300", "value": "true", "keyword": ""},
            {"id": "orientation_starter", "volatility": "300000",
            "update": "100", "value": "true", "keyword": ""}],
        "local_links": [
            {"from": "nfc_starter", "to": "nfc"},
            {"from": "nfc", "to": "phone_plugger"},
            {"from": "orientation_starter", "to": "orientation"},
            {"from": "orientation", "to": "phone_plugger"},
            {"from": "position_correction", "to": "hud"}],
        "out_links": [
            {"from": "nfc", "to": "screen_connector",
            "device": "screen"},
            {"from": "phone_plugger", "to": "screen_connector",
            "device": "screen"}],
        "in_links": [{"from": "leap", "to": "position_correction",
            "device": "leap"}]},

        {"device": "screen",
        "components": [
            {"id": "screen", "volatility": "600000",
            "resource": "/scripts/sensor/screen.txt"}],
        "nodes": [...],
        "starters": [],
        "local_links": [
            {"from": "screen_connector", "to": "angle_correction"},
            {"from": "angle_correction", "to": "angle_convertion"},
            {"from": "angle_convertion", "to": "angle_sincos"},
            {"from": "angle_sincos", "to": "angle_matrix"},
            {"from": "angle_matrix", "to": "screen"}],
        "out_links": [],
        "in_links": [
            {"from": "nfc", "to": "screen_connector",
            "device": "phone"},
            {"from": "phone_plugger", "to": "screen_connector",
            "device": "phone"}]},

        {"device": "leap",
        "components": [
            {"id": "leap", "volatility": "120000",
            "resource": "/scripts/sensor/leap.txt"}],
        "nodes": [...],
        "starters": [{"id": "leap_starter", "volatility": "300000",
            "update": "100", "value": "true", "keyword": ""}],
        "local_links": [{"from": "leap_starter", "to": "leap"}],
        "out_links": [{"from": "leap", "to": "position_correction",
            "device": "phone"}],
        "in_links": []}
    ]}
]}
```

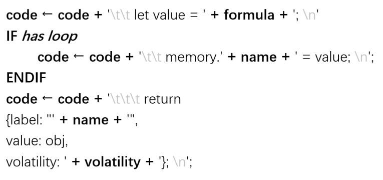
Figure 7. The JSON configuration for the MusExp App with the two applets for the smartphone device merged.

Analyzing the JSON configuration file for the MusExp application, we see that there are three devices involved: the smartphone performing two distinct roles in the two applets, connecting as an input device for the screen and as an output device for the LeapMotion controller. The special nodes are listed in the components field and contain written JavaScript code. The starter nodes are specified in the starters field, whereas JSON configurations for the processing nodes designed through the no-code visual interface would be listed as an array in the nodes field, and two examples may be visualized in Figures 8a and 9a. Links of information flow are grouped into three categories, the in and out links connected to the Socket.io server, and the local links consisting of direct function calls between code components present on the same device. When a message is received from an external source through the Socket.io server, after it is parsed the communicator also directly calls the internal-processing function of the specific local node module.



```
{"name": "acceleration_total",
"type": "numerical",
"memory": [
        {"label": "acceleration"}],
"loop": "false",
"root":
        {"input": [
                {"label": "acceleration", "timeout": "false"}],
        "output":
                {"label": "acceleration_total", "value": "Math.sqrt(
                Math.pow(memory.acceleration.value.x,2) +
                Math.pow(memory.acceleration.value.y,2) +
                Math.pow(memory.acceleration.value.z,2)
                )", "volatility": "1000"}}}
```

(**a**)



```
code ← code + '\t\t let value = ' + formula + '; \n'
IF has loop
    code ← code + '\t\t memory.' + name + ' = value; \n';
ENDIF
code ← code + '\t\t\t return
{label: "' + name + '",
value: obj,
volatility: ' + volatility + '}; \n';
```

(**b**)

Figure 8. (**a**) The JSON configuration for the total acceleration magnitude calculator node that uses feedforward numerical processing. (**b**) The pseudocode of the server components that compiles the numerical-processing JSON file delivering JavaScript code.

```
{"name": "daylight_classifier",
"type": "nominal",
"memory": [
        {"label": "sun_altitude_calculator"}],
"loop": "false",
"root": [
        {"input": [
                {"label": "sun_altitude_calculator", "value": "< -10", "timeout": "false"}],
        "output":
                {"label": "daylight_classifier", "value": "'night'", "volatility": "600000"}},
        {"input": [
                {"label": "sun_altitude_calculator", "value": "< 0", "timeout": "false"}],
        "output":
                {"label": "daylight_classifier", "value": "'twilight'", "volatility": "600000"}},
        {"input": [
                {"label": "sun_altitude_calculator", "value": "< 10", "timeout": "false"}],
        "output":
                {"label": "daylight_classifier", "value": "'flares'", "volatility": "600000"}},
        {"input": [
                {"label": "sun_altitude_calculator", "value": ">= 10", "timeout": "false"}],
        "output":
                {"label": "daylight_classifier", "value": "'day'", "volatility": "600000"}}]}
```

(**a**)

```
code ← code + '\t\t let xor = function(a,b) {return ((a && !b) || (!a && b));} \n'
code ← code + '\t\t let time = new Date().getTime(); \n'
FOR each category
    code ← code + '\t\t if ('
    FOR each condition
        code ← code + '(memory.' + label + '.value' + test + ')
        && xor(time < memory.' + label + '.timeout , ' + flag + ')'
        IF not last
            code ← code + '&& '
        ENDIF
    ENDFOR
    code ← code + ') { \n'
    IF has loop
        code ← code + '\t\t memory. ' + name + ' =
        {value: ' + value + ', timeout: (time + Number(' + volatility + "))}; \n'
    ENDIF
    code ← code + '\t\t\t return
    {label: "' + name + '", value: ' + value + ', volatility: ' + volatility + ' }; } \n'
ENDFOR
```

(**b**)

Figure 9. (**a**) The JSON configuration for the daylight-classifier node that uses feedforward nominal processing. (**b**) The pseudocode of the server components that compiles the nominal processing JSON file delivering JavaScript code.

4.2.2. Application Builder Compiler

We devised two types of JSON configurations, with a third one created by merging those two, that delivers for non-programmers some extended capabilities. We are first going to show an example of a formula-based processing node and a decision-based processing node, as JSON configuration files and server-side code presented in pseudocode, that interprets the configuration and produces JavaScript code.

The first JSON configuration type that we are going to explain is the numerical processing node presented in Figure 8a and previously illustrated in Figure 4a in relation to the graphic user interface. The JSON file contains the name and type of the node; the list of nodes that this node is listening to and those from which it is saving data in the memory field; whether the node has a loop, meaning that its output becomes an input to itself; and the root of the processing operations, which in this case helps compute the 3D magnitude of the acceleration vector from its orthogonal components. The server-side code is shown in Figure 8b, and it generates the module for this node which, is delivered to the device. The code assembles the calculation formula, saves the output in the memory along with other inputs if the node has a feedback loop, and adds the return expression, which outputs an object with a label, a value, and a volatility field.

The second type of JSON configuration that we will discuss is the nominal processing node, which we previously presented in terms of the graphic user interface in Figures 4b and 5. The JSON file in Figure 9a specifies the name of the module that corresponds to the processing node and the type, which in this case is nominal so that the compiler knows how to interpret the code. The memory field specifies which nodes send information to the present node in order to save the latest input data. In this case, there is only one node, but when there are multiple inputs, in the current version of the platform there is an output computed for each new input received, considering that the other memory fields have not yet reached timeout. The loop field, which refers to the oriented graph information flow logic, specifies whether there is also a memory field for the output produced by the current node. In the ConfiDrive App, the node functioning as the driver's stress classifier, which uses feedback decision-based processing and models a state machine for inferring the level of stress or tiredness of the driver, has four memory fields and a loop field with the Boolean value of "true." Although the loop-enabled node is more complex, the compiler interprets the node in the same manner, and the user creating the application works at the same level of complexity from the graphic user interface. In Figure 9b the server-side pseudo-code that interprets nominal-node JSON configuration files is shown. The compiler creates a series of IF statements consisting of chained conditions, consisting of value tests and timeout tests, followed by return statements.

For increasing capabilities, a combination of numerical and nominal processing nodes was devised, which we will call the general node. In Table 2, the three types of nodes are presented, and we should state that the general numerical and nominal output nodes are subsets of the general node. The general node takes the complexity of input from the nominal node that uses IF statements and the complexity of output from the numerical node that uses formula statements. The numerical node is a subset of the general node because its input may be expressed as an "IF (true)" statement followed by the complex output formula statement. The nominal node is a subset of the general node and may also be seen as having a plain formula of output that consists of a primitive value representing the classified category. The extended capabilities of the general node are useful in filtering data and deciding which formula to use for computation. Thus, each output category from the IF statement may use different formula statements or produce no output at all. More complex decisional structures may require rule-managing algorithms such as dynamically auto-compiling an optimal decision-tree structure from the listed rules, with ordering based on the truthfulness probability of the IF statement and the execution-speed requirement of the THEN statement.

Table 2. The three types of nodes as interpreted by the compiler, with the first two being particular cases of the third node.

Name	Input	Output
Numerical node	Simple (plain data/object)	Complex (formula statement)
Nominal node	Complex (IF statement)	Simple (plain data/object)
General node	Complex (IF statement)	Complex (formula statement)

We present the general node in action implemented inside the connectivity pattern, presented in Figure 10, for multiple users connecting to the MusExp Application. The setup constitutes of a widescreen that displays an artifact, and each museum visitor who scans the NFC tag connects their phone to the screen and may rotate and visualize the artifact. This configuration permits the user who scans the NFC to disconnect the previous user in the following manner. When scanning the NFC tag with the help of the NFC sensor node present on the smartphone device, the user sends to the screen the tag information along with the scan timestamp. The screen then saves these data inside the memory field of the screen-connector node. At the same time, the NFC sensor node present on the smartphone sends the tag and timestamp information to the phone-connector node also present on the smartphone, and this node saves the information inside the memory field. When the orientation-sensor node reads data from the smartphone, it first concatenates these data with data from the NFC tag information and timestamp, producing an object, which it sends to the screen-connector node present on the screen device. If the NFC information and timestamp of the smartphone coincide with the data present in the memory of the screen, the screen device listens to this smartphone, whereas if other devices are sending information that does not coincide, the screen ignores that information. Thus, a connectivity pattern is achieved with this type of general processing node, which in this case is the screen-connector node. The phone-connector node is of the numeric processing type, concatenating data into an object following a formula statement.

In Figure 11 we present the graphic user interface for creating applications. The left navigation bar opens right next to the list of items that may be enabled to appear in the vertical icon menu. Categories of items are hardware devices involved in the distributed application, the connectivity patterns between those devices, input nodes and processing node templates, output nodes, and the corresponding multimodal assets. By enabling the smartphone as hardware, we may proceed to enable the smartphone orientation sensor and the smartphone NFC sensor, which appear as deep-blue icons in the menu. Next to the menu, there is a panel for constructing the node properties. We present the configuration of a formula-based feed-forward processing node involved in adjusting the alpha-angle magnitude to be mapped correctly between the orientation of the smartphone and the 3D virtual object presented on the wide screen. When the node is ready, it may be dragged and connected to the multimodal interaction MusExp app present on the right side of the panel. By clicking the collapsed menu icon of the hardware we may visualize the nodes, and when nodes are clicked they expose the internal configuration on the left panel. The top navigation bar includes other user roles, such as the explorer, the developer, and the owner, which provide search options and configuration tools according to the targeted attributes for each category.

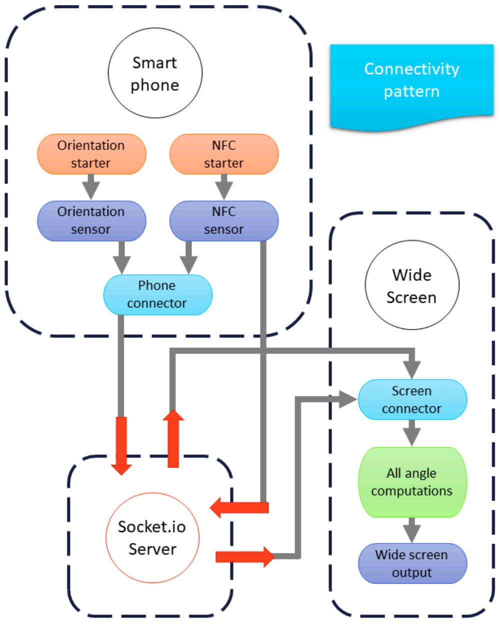


Figure 10. The connectivity pattern achieved with the help of the general node and an NFC scan event, which helps users connect one at a time to the MusExp application for interacting through their smartphone with the artifact displayed on the widescreen.

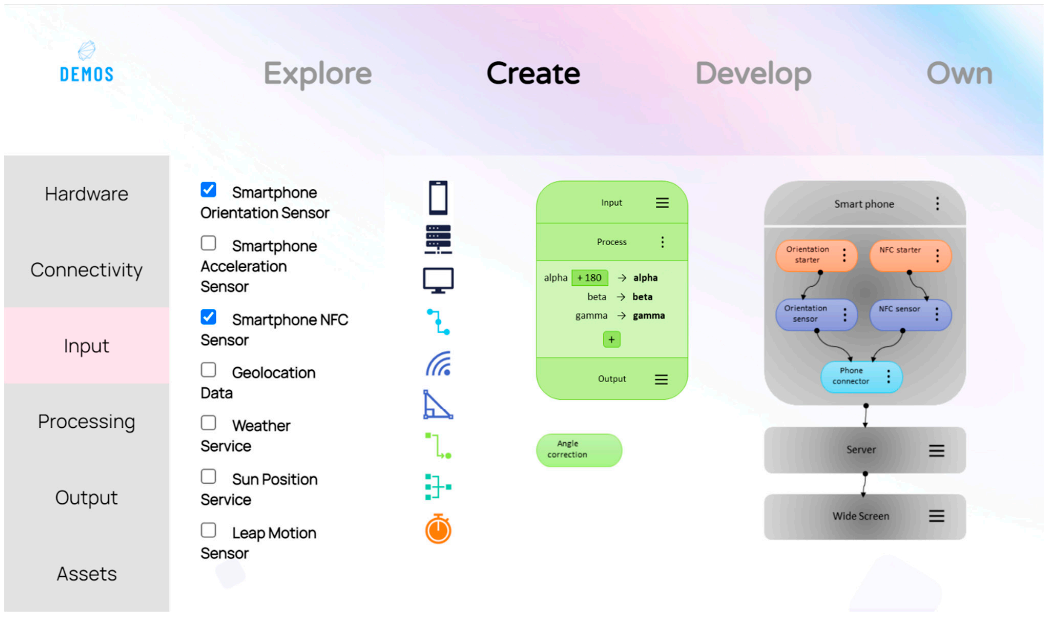


Figure 11. The graphic user interface when creating the MusExp app for multimodal interaction with the connectivity pattern for disconnecting the previous user.

4.3. The Core of the Concept and the DEMOS Platform

The conceptual scheme of the DEMOS platform is shown in Figure 12 and is horizontally divided into four levels. At the Logic level, there is the general node (see Table 2), which represents the formal generalization of the numeric and nominal nodes, and using the general node we provide the creators with several connectivity patterns (see Table 1). The Configurations level (see Section 4.2.2) refers to the two JSON file structures corresponding to the decisional and formula statements that are interpreted by the compiler present on the server to produce modular JavaScript code for each device part of the distributed application. The Concepts level comprises the intuitive no-code templates for processing nodes that help naïve users create applications. The Interface level offers the hands-on tools for applet creation by non-programmers and contains multiple emerging features.

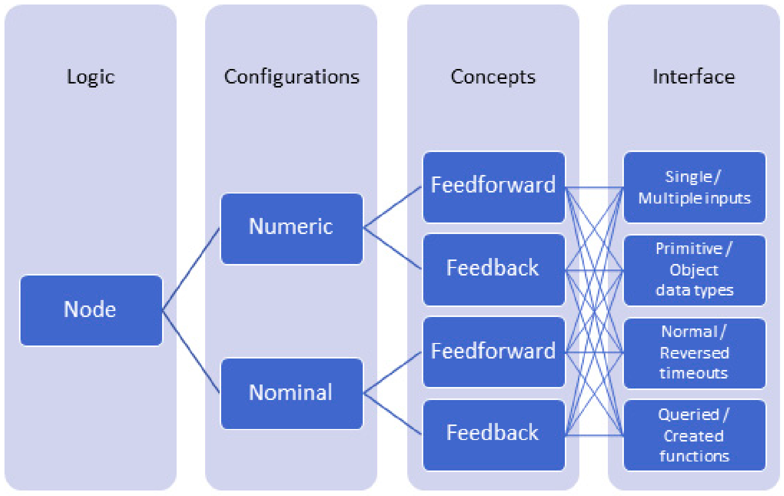


Figure 12. The conceptual scheme of the platform starts from the Logic level of the general node present on the server-side compiler, the JSON Configurations level corresponding to the decisional and formula statements, and the Concept level of nodes in the intuitive no-code templates and ends with the Interface level, which offers the hand-on tools for applet creation by non-programmers.

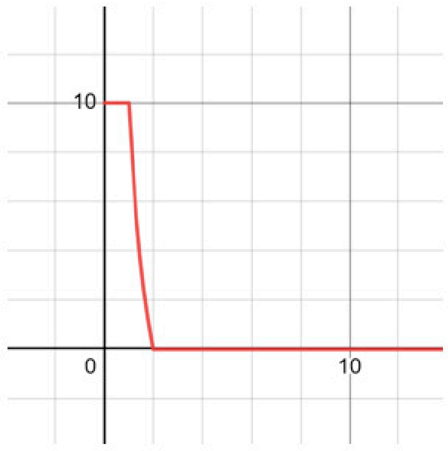
In this section, we will tackle the Concepts level. The fully configurable nodes of the platform are the core nodes used for processing data that flows between the input nodes and the output nodes. We separated them into two categories according to their output, the first one being the numeric processing nodes and the second being the nominal processing nodes. The numeric processing nodes usually take numeric data and input it into a formula that produces a numeric output, whereas the nominal processing nodes label both numeric and nominal data that satisfy a certain logic configuration. These two categories are further split into feedforward nodes and feedback nodes. The feedforward nodes take one or more inputs and create an output, whereas the feedback nodes are similar in purpose to the above category, but they are more complex in that they combine the internal value of their previous output with that of the input coming from the other nodes. The feedback feature helps build the logic of automatons and recursive formula structures.

Although numeric processing nodes and nominal processing nodes differ in the way the JSON configuration looks and is interpreted and transformed into real JavaScript code on the server side, feedforward and feedback nodes are handled the same way by the interpreter. They only provide a distinct programming logic for the person who creates the applet, and specifying the output of the node as an input field is enough to transform a node from the simple feedforward configuration to the more complex feedback configuration.

4.3.1. Formula-Based Processing (Numerical Node)

This type of node is intended for mapping continuous values using mathematical formulas. Examples include feedforward processing for mapping Euler angles to quaternions, mapping different color scales between one another, such as RGB and HSV; mapping different numerical ranges between each other, such as LeapMotion 3D coordinates to Three.JS coordinates; and calculating the 3D-vector magnitude with Pythagoras' formula from the orthogonal values. They may be used for visualizing data in innovative ways and for creating interaction metaphors. They may also keep track of their previous outputs by using them in the more complex form of feedback processing nodes.

The configuration of this type of processing node consists of specifying the input fields, the formula, and the output values, where the input values may also contain the previous output of this exact node. It is implemented by processing the JSON configuration file, which may contain more than one input field and a single output field. In the wider sense, numeric processing is formula-based processing that may output objects containing concatenated numeric or non-numeric data processed through calculations, Boolean operators, and string operators. Thus, the numeric processing node aids in collaborative applications combining data coming from several users' input devices. For example, we may imagine three users that are only able to move a virtual heavy object by coordinating their actions, namely, their hand movement while holding their smartphone. In Figure 13 we present a formula that computes the difference in the X direction of acceleration between three users taken two by two and then calculates the magnitude of the teamwork with the help of a formula. The accelerations were previously filtered so that the action is carried out only when the smartphone devices are nonstationary. We see that as x approaches 0, meaning that the coordination is almost perfect, the magnitude of y is 10 and the artifact is moving, otherwise, y drops quickly towards 0 and the artifact suddenly stops moving.



$$\max\left(\frac{20}{\max(x, 1)}, 10\right) - 10$$

$$x = |x_1 - x_2| + |x_2 - x_3| + |x_1 - x_3|$$

Figure 13. The graphic and the formula used for three users to collaboratively move a virtual artifact only when they coordinate their acceleration in the x-direction with a certain precision.

4.3.2. Decision-Based Processing (Nominal Node)

This type of node is intended to classify numeric or non-numeric values and label categories in the form of feedforward processing. It may also function as an automaton with internal states in the form of feedback processing. Simple examples include classifying acceleration into low and high, classifying labels of weather phenomena into good and bad, or labeling sun altitudes above and beneath the horizon into daylight and night. The feedforward processing node may also help design a tour guide in which certain triggers such as reading NFC tags may enable outputs or furthermore may enable features and distinct sub-applets. In the wider sense, nominal processing is decision-based processing that may output plain data or objects based on a logical test. Thus, we can also mention the more complex examples of feedback nominal processing that can handle more powerful logic, such as predicting the arousal state of the driver using time-dependent decisions triggered by external stimuli with certain timeouts attached to each event.

$$\text{XOR }((\text{timeout} > \text{present}), \text{inverse\_timeout\_flag}) \qquad (1)$$

The configuration of the nominal processing node consists of specifying the logic using a multi-case layout. Each case may have one or more inputs and a single output. The inputs contain a logical test against either a numeric value or a non-numeric value and a time-related test. Non-numeric values may be tested for equality or difference, whereas numeric values may be compared using the mathematical operators. In the current version of the platform, the AND operator is placed between the array of conditions listed for each case, and the OR operator is represented by writing two distinct cases of conditions with the same output. The time-related logic component tests the time of the produced output for a timeout against the present time and adds an XOR operator for the result and a Boolean value specified in the configuration, as presented in Expression (1) in terms of code logic. This particular logic component offers the possibility to go into certain states while the input is still valid timewise or to go into certain states after the input has reached timeout.

5. Testing the DEMOS Prototype Platform

The platform was used in order to create and explore two applications for distinct contexts. The two applications were presented throughout Section 4, focusing on elements serving as proof-of-concept for the prototype platform. Several adjustments to the applications presented in Figures 3b and 6 were also discussed in order to exemplify specific capabilities and logic. This particular choice of application permitted experimenting with additional concepts, which will be presented in the following section. The core of the applications remains the one shown in the previously mentioned figures since it perfectly embodies the main concepts of this article.

Our first demonstrative applet, named ConfiDrive, is a single-device applet, whereas the second one runs on a distributed hardware configuration and is named MusExp. The first applet (see Figure 3b and Section 5.1) is designed to be used inside a car for safe driving, whereas the second one (see Figure 6 and Section 5.2) is designed to be used in a museum context for cultural and entertainment purposes. The first applet mostly relies on decision-based processing with nominal output, whereas the second one consists of interactive components mapping numerical data by formula-based processing. We showed both groups of non-programmer and programmer students the two applications developed using the platform—the ConfiDrive App and the MusExp App—and asked them to rate the applications over a five-fold scale of several user-experience dimensions, as detailed in Section 5.3.

5.1. ConfiDrive Architecture

The first applet that was created using the platform is purposed to be used inside a car while driving and has the aim of estimating the external stimuli affecting the driver both by cognitive and sensory overloading and deprivation. It is a standalone mobile application (see Figure 3b) inspired by a previously designed system [6] that was more complex, having components for visualizing information and multimodal interaction. By selecting the core components of this system and implementing them within the platform, we tested the platform's capability for creating applets and proved a first use case of the system.

This applet was constructed to be run on a single Android device with internet service. The input nodes for information are the geolocation service, which uses the Geolocation API (Geolocation API - Web APIs | MDN. Available online: https://developer.mozilla.org/en-US/docs/Web/API/Geolocation_API, (accessed on 23 June 2022)); the weather service component, which fetches information from the Open Weather API (Weather API - OpenWeatherMap. Available online: https://openweathermap.org/api, (accessed on 23 June 2022)); and the sun-position component, which calculates the sun's altitude using the astronomical equations from the SunCalc library (GitHub - mourner/suncalc: A tiny JavaScript library for calculating sun/moon positions and phases. Available online: https://github.com/mourner/suncalc, (accessed on 23 June 2022)); as well as data from the device's accelerometer without the gravity vector (DeviceMotionEvent.acceleration - Web APIs | MDN. Available online: https://developer.mozilla.org/en-US/docs/Web/

API/DeviceMotionEvent/acceleration, (accessed on 23 June 2022)) by capturing the device-motion event from the window object.

A timer component notifies the geolocation component to send location data to the weather and sun components. Then the weather component sends data to a feedforward nominal processing node that classifies and labels weather as either good or bad. The sun component also sends data to a feedforward nominal processing node that interprets altitude degrees as daylight or night. There is also a timer component that tells the acceleration component to read data and send it forward first to a feedforward numeric processing node to calculate the 3D acceleration magnitude with the help of Pythagoras' formula and then to a feedforward nominal processing classifier node that labels measurements into either low or high values.

The data travel forward to a feedback nominal processing node that combines the latest data from itself and classified acceleration, weather, and natural lighting data. If the weather is bad or the acceleration is high, then the output is overloaded by stimuli and the state timeout is 5 min, which may be extended through renewal by an identical input. If this timeout passes without renewal and the internal state is still overloaded by stimuli, then the output changes to the normal load of stimuli, with a timeout of 5 min and no extension possibility because of the logic. If this longer timeout passes and it is night, then the output changes to being deprived of stimuli. If it is daytime, then after a timeout the system goes into the default state, which is the normal load of stimuli. The end output node may be using an audio component that plays either relaxing music for an overload state or an alert for a state of deprivation. For this demonstrative application, we used two text-to-speech phrases, "Relax" and "Freshen up," voiced by a female-speech synthesizer (Voice Generator (Online & Free). Available online: https://voicegenerator.io/, (accessed on 23 June 2022)).

5.2. MusExp Architecture

The second applet that was created using the platform is intended to be used inside a museum to enhance the cultural experience of the visitors. It is inspired by previous work with means of digital interaction and dissemination of archeological heritage [57] and a ubiquitous intelligent-museum environment [5]. This applet proves the capability of the system to handle distributed interactions over a series of different smart devices.

This application runs on an Android smart mobile phone, a computer connected to a LeapMotion controller (Tracking | Leap Motion Controller | Ultraleap. Available online: https://www.ultraleap.com/product/leap-motion-controller/, (accessed on 23 June 2022)), and a computer connected to a widescreen. The application is composed of two main applets (see Figure 6), namely, the multimodal interaction applet and the XR visualization applet, with an additional one based on triggering audio content by scanning tags. An NFC tag-reader component based on NDEFReader (NDEFReader - Web APIs | MDN. Available online: https://developer.mozilla.org/en-US/docs/Web/API/NDEFReader, (accessed on 23 June 2022)) helps enable applications and provide methods of connectivity for multiple users, as mentioned in Table 1. This component is triggered by a timer component, and forwards data into a feedforward nominal processing node that classifies the last read tag with one of the following labels: rotate, illuminate, or narrate. The NFC tags help enable certain components of the system as follows.

The first tag enables the interaction between the mobile phone's rotation sensor and the computer connected to a widescreen. The mobile phone starts to continuously send Euler-angle values captured by the device-orientation event (Window: deviceorientation event - Web APIs | MDN. Available online: https://developer.mozilla.org/en-US/docs/Web/API/Window/deviceorientation_event, (accessed on 23 June 2022)). These are forwarded to a feedforward numerical processing node in order to be mapped to quaternion values. This component then sends data through the server to the computer connected to a widescreen, where the end-output component rotates an artifact in real time as the user orients their phone in mid-air. We should mention that on the horizontal plane, the screen is aligned towards the north, as the angles are calculated in relation to the Earth's magnetic field.

The second tag enables interactions between the virtual scene displayed on the mobile phone and the computer connected to the LeapMotion controller. The mobile phone asks for permission to go into full-screen mode and an artifact is placed on the screen in dim light. While the phone is placed beneath an HUD in a dark room, the LeapMotion controller captures the user's hands in a mid-air position above the hologram that appears. The computer sends real-time data of the index fingers to a feedforward numerical processing-mapping component and then through the server to the mobile phone, where Three.js PointLights (three.js examples. Available online: https://threejs.org/examples/?q=pointligh#webgl_lights_pointlights, (accessed on 23 June 2022)) are placed accordingly in 3D space. This gives the appearance of a 3D object being lit by the user, who is supposedly holding the light with their fingertips. We also experimented with coloring the light by implementing a feedback numerical processing node that tracks the hands' positions and adds to the current value on a mapped HSV scale the distance from the previous point in space. The effect is that the color changes as fast as the hands move above the object.

Another applet for the museum experience narrated information about the artifact by reading multiple NFC tags and enabling resources of sound output on the smartphone device. The tags may be spread throughout the physical space for anchoring applications and information for an interactive and digitally enhanced museum smart tour. In this case, audio information was played on the phone after reading the tag in order to guide the visitors and provide additional information related to using the multimodal interactive spots.

5.3. Application Rating after Experiencing the DEMOS Prototype Platform

We presented to the students two application schemes, the corresponding JSON files, resources for the applications, the code generated by the platform, and the actual application running on hardware. We let them grasp the modularity and features of the system, and then we let them try each application hands-on. We asked for feedback regarding usefulness, pleasantness, and innovation, which we measured on a 5-point Likert scale.

The results from Figure 14 show that all respondents rated the apps between 1 and 2 on a 5-point Likert scale (−2 = very negative, −1 = negative, 0 = neutral, 1 = positive, 2 = very positive, with respect to each of the three traits: usefulness, pleasantness, and innovation), and there were no answers corresponding to the negative side of the scale. The usefulness of the ConfiDrive app slightly overrode that of the MusExp app, with the users selecting either "useful," or "very useful" and one "so and so" for the museum app. For the second user experience dimension, the MusExp app received significantly more points for pleasantness, this time with most of the answers corresponding to the "very pleasant" option. As for perceived innovation, both applications scored identically on the scale and had the same amount of each answer choice, with seven students answering "very innovative," four responding with "innovative," and only one with "so and so."

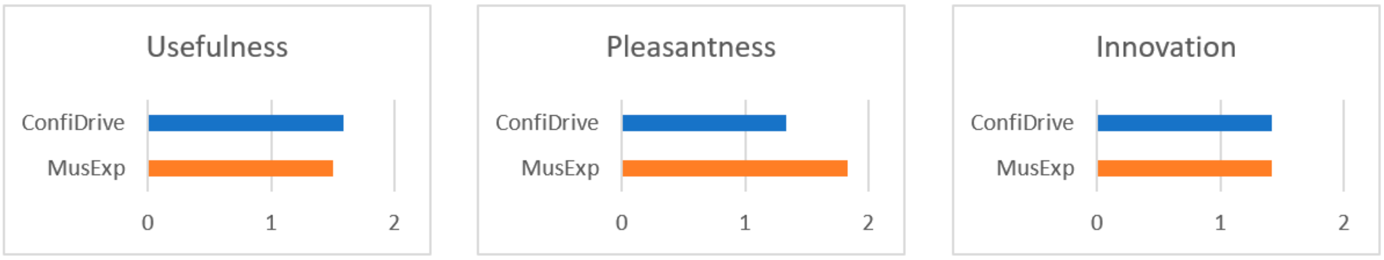


Figure 14. Results after testing for the perceived usefulness, pleasantness, and innovation of the ConfiDrive and MusExp apps, plotted on a Likert scale where 0 is neutral, 1 positive, and 2 very positive.

Overall, there were only slight differences between the responses in terms of study domain, sex, and age. The female students seemed to choose higher scores for pleasantness and male students for usefulness. In addition, in terms of innovation, the non-programmers seemed to score higher by choosing the high-end option on the scale more, whereas programmers were divided along the positive side of the scale with two neutral responses. In this small sample, age did not seem to be a factor in rating the applications differently.

6. Testing the Conceptual Framework

We tested the conceptual framework of the DEMOS platform in order to assess the intuitiveness of application-development patterns for naïve users and to devise a future development roadmap in order to create an engaging platform-based sustainable digital ecosystem. First, we asked students to imagine one IFTTT application and one more complex application that requires intermediary processing or converging inputs in order to check the affordance of the concepts and patterns (see Section 6.1). We let the non-programmer students work on a team and the programmer students to work individually and compared the results. Second, we asked the same students what the most important features of such a platform are in order for them to engage with the system (see Section 6.2).

There were two testing groups, a group of non-programmers coming from scientific and humanistic study domains and a group of computer science students who know how to code, with each of the groups made up of six students. In total, we received feedback from five female and seven male students aged 22–32. The study domains of the non-programmers were medical science, veterinary science, sports, mathematics, psychology, sociology, and arts. The summary of the testing manners corresponds to the following sections.

6.1. Experiment for Assessing the Platform's Concept and Pattern Affordance

In the second testing scenario, we asked the students to imagine applications based on their understanding of the concepts on which the platform is based. We asked them to list some interesting data inputs from smart sensors or web services, and some outputs consisting of either automating smart devices or generating multimodal XR visualizations. Then, we asked them to state what kind of processing steps need to be implemented between the inputs and the final output.

The computer-science students managed to imagine applications that are both useful and may be implemented using the core visual-programming templates of the platform. First, we will give some examples of the IFTTT apps that require more programming on the input and output sides compared to the processing nodes. These include an app to estimate meeting times for two people and produce notifications or use geofences for knowing when family members have left their stationary whereabouts, requiring intelligent-input handling. Another exemplified IFTTT app was the personalization of smartphones in terms of sounds, appearance, and apps listed on screen by creating profiles depending on geolocation, tags, and other detected events, requiring special access and a diverse array of outputs. One more app that we will mention provides suggestions for spending leisure time and will both require special inputs and outputs in order to accomplish the task of a personal assistant.

Apps mostly relying on processing nodes are a perfect fit for the platform. They easily reuse special nodes, and the intermediary processing work is carried out with the help of the implemented available templates. A good example devised by programming students involved an adaptive city tour that provides useful insight and guidance for travelers and locals alike. Several other applications involved home, office, classroom, and gardening automation, which require nominal and numerical processing for categorical labeling and quantity mapping purposes. Another related application tackled home-environment modification using a natural intelligent interface that maps gestures and produces smart effects. Lastly, there was an app that targeted transportation in a smart city, and it involved waiting times and congestion relief.

Although it was expected that computer-science students would be able to productively use the concepts behind the platform, we focused on the non-programmers, who were inclusively integrated into the developer's horizon of possibilities. The applications that were proposed by non-programmers are listed below in Table 3. The first column states the purpose of the application. The second describes the special nodes designed by programmers for input, output, and special tasks. Within the third column is the central concept of processing nodes that justifies the design of the platform as an environment to air non-programmers in application development. The last column contains comments that explain how the student adopted the vision, used the templates, and states eventual logical flows and future necessary features to develop. Overall, for first and short contact with the conceptual framework, the non-programmer students performed well and most of the time succeeded in designing functional applications.

Table 3. Applications that are imagined and designed by non-programmers.

Application Description	Code Resources	Processing Nodes	Comments
Application for notifying you a few hours before the rain will start	▪ Weather-service input ▪ Sound output for notification	▪ No nodes ▪ (feedforward nominal decision)	The application was first confused with an IFTTT app with no processing. There should be a decisional node for classifying weather. The application is a good fit for the platform.
Application for organizing school billboards and wall art according to your school class, current events, and personal preferences	▪ Class input ▪ Preferences input ▪ Location input (QR code/NFC tag) ▪ Visual output for events	▪ Feedforward nominal decision	The application is a good fit for the platform, using the implemented template for no-code programming.
Application for notifying your partner with vibrations on a smart bracelet/jewelry that you are trying to reach them while their phone is muted	▪ An input button to tap ▪ The bracelet's output method	▪ No nodes	The application is a simple IFTTT app that may be also designed with current platforms on a smartphone. The smart device needs to be browser compatible with functioning on the current architecture.
Application for donating money to a cause in an amount directly proportional to screen time on social-media apps	▪ An input for screen time (an automatic input may be done by scanning an NFC tag or navigating the web from within a future platform) ▪ An output as money donation (a future platform shall be linked to a wallet)	▪ Feedforward numerical mapping screen time to money	The application is a good fit for the platform. The screen-time-measuring method was not included in the logic, but it may be done by accessing social media with an NFC scan and scanning another tag to stop monetizing. A future platform should be bound to a digital wallet.

Table 3. *Cont.*

Application Description	Code Resources	Processing Nodes	Comments
Application for a smart refrigerator that alerts the user of approaching expiry dates and low quantities of staple food	▪ An OCR input for scanning dates (or future QR codes containing such info) ▪ Several weight sensors ▪ A visual on the fridge ▪ A visual output on the phone	▪ A feedforward node for each weight sensor storing date after a scan ▪ A feedforward checker of expiration and weight	The application is a good fit for the platform. It was said to be inspired by the pattern of connecting the mobile device to the screen in the museum application. Here a date is merged with the initial weight and then constantly checked.
An application for house or private-property safety against burglars or stalkers who stay still for more than 5 min	▪ A camera with computer vision for determining peoples' position ▪ An alert for the owner ▪ A scaring noise for the presumed burglar	▪ A feedback numeric node measuring a person's moving distance ▪ A feedforward nominal node discriminating between still and moving	The application is a good fit for the platform. Another pattern exposed in the presentation was implemented, which is slightly more complex in requiring a feedback loop.

6.2. Survey for Characteristics of a Platform-Based Sustainable Digital Ecosystem

The final step of the testing is requiring a grasp of the essence of the conceptual platform that is creating an impact on sustainable development within the greater digital transformation of society. We thus first asked the students to analyze the most important characteristics of the platform that assure its adoption, and respond with "essential," "relevant," or "irrelevant."

In Figures 15 and 16 we list in descending order the most desired characteristics for programmers and non-programmers, respectively, in order to get involved with the platform. Overall, the reuse of resources and programming templates scored the highest for both groups, followed by anchoring apps, micropayments, and innovation. Expected differences arose from the fact that non-programmers were unable to contribute to the platform's open-source core components or to the nodes that require programming skills but were eager to use no-code visual programming in order to implement ideas and contribute to innovation. Other observations are that non-programmers were very excited about anchoring apps and services for real-world locations and events. Compared to programmers, they scored higher for interpersonal interaction and supporting causes and creators, whereas programmers liked the idea of living in a digitalized world. Additional personally listed desired characteristics included owning and control of personal data and democratic voting for future platform developments.

We asked the students which domains need a digital transformation and may benefit from the developed prototype platform that enables people to visually create apps. A domain that was mentioned twice was the development of a circular economy with microservices for recycling that both helps incentivize positive action and helps provide local data for implementing sustainable-recycling awareness, policies, and mechanisms. One answer regarded social innovation in order to build new ways to find and interact with volunteers, experts, and collaborators. Another domain that also was mentioned twice was the health sector, which may benefit from augmented and multimodal consultations for doctors and psychologists in a process that permits a more holistic approach, and the wellness sector of fitness and meditation, which would provide people with collective and enriched experiences. Other mentioned domains that are also connected to human health and the environment were smart shopping and smart fridges in order to support countering food addiction and food waste. Energy-efficient and safe homes and transportation were other areas where we need apps for automatically turning on and off sockets or apps such as XR optimized navigation maps.

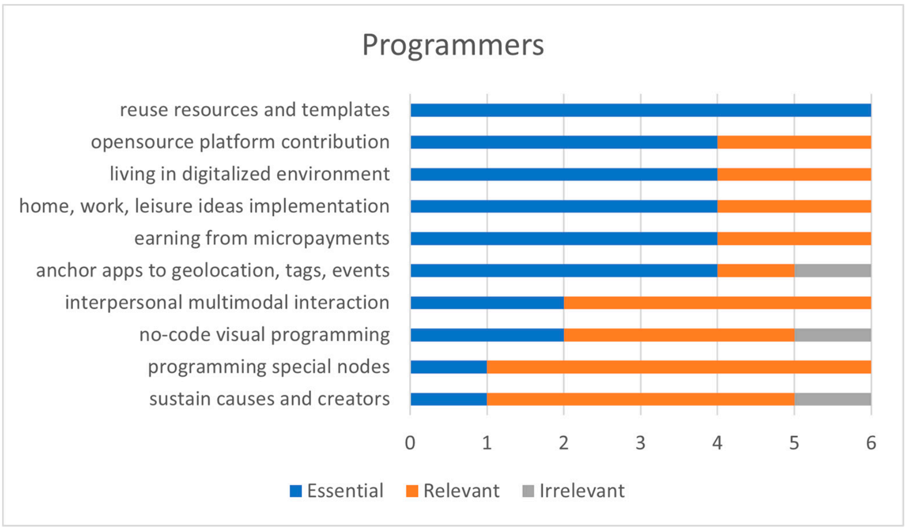


Figure 15. Most desired characteristics by programmers in order to get involved with the platform.

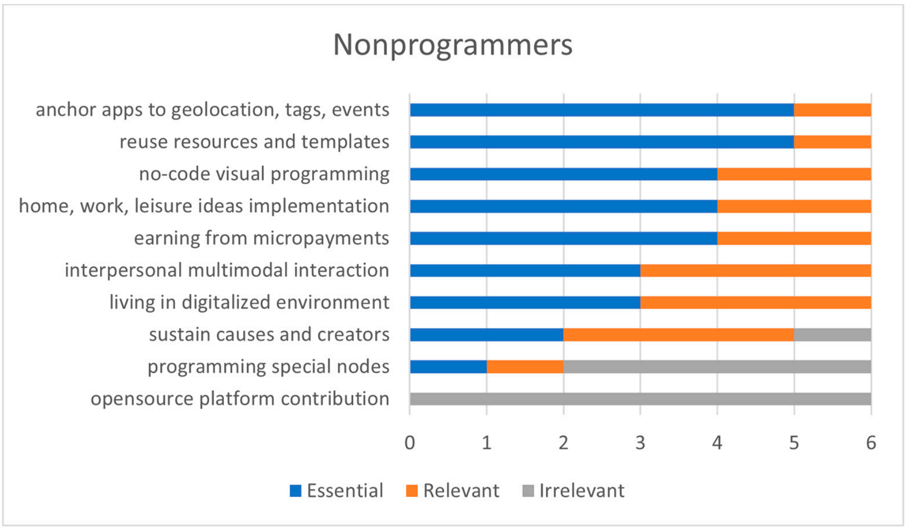


Figure 16. Most desired characteristics by non-programmers in order to get involved with the platform.

Students concluded that the existing platform may contribute to positive social change through digitalization, solving some key problems and bringing value to people's life experiences. Students affirmed that the platform successfully extends the capabilities of non-programmers to build applications and helps programmers quickly implement ideas as prototypes to receive feedback from the targeted audience. The respondents looked forward to adding elements of development sustainability to the platform, such as data security and digital currency. Students asserted that the platform is a good fit for a tool to create an intelligent environment enhanced with MR and IoT applets. To overcome the current limitations of the study sample, it is beneficial to explore the needs of non-programmers coming from a more diverse set of social categories such as urban and rural residents of different age groups, socio-economic classes, and cultural backgrounds.

7. Conclusions

This article presents a conceptual prototype platform for designing intelligent environments within the larger context of sustainable development by bridging the digital divide across different dimensions. The platform offers intuitive tools for non-programmers to build applets with the help of resources made available by programmers who write code for reading smart-sensor data and web-service information.

With no-code visual-programming tools, users create formula-based and decision-based processing nodes and link them together with the nodes provided by programmers, designing the oriented graph for the flow of information that constitutes the core of the application. Templates of connectivity and interaction along with the configuration logic of the processing nodes are available to help implement novel ideas or adapt existing applets to the new context.

A Node.js Express server interprets configurations designed by the users and delivers the JavaScript code of processing nodes for each device part of the distributed application. The information-flow scheme between the modular parts is provided on device initialization by the Socket.io server, which also offers a means of communicating events between devices. The programmed special nodes provide multimodal interaction with elements of XR and the IoT.

The prototype platform was tested by building two applications for different contexts. The first was a standalone application for safely driving a car by inferring the cognitive load of the driver using acceleration, weather, and natural-lighting data, preventing falling asleep at night and providing relaxing feedback when the situation is tense. The second application, consisting of two main applets, enhances the museum experience for visitors. Through multimodal interaction, a visitor rotates an artifact displayed on the museum's screen with their phone, and in a dark room the mobile phone displays an artifact that is virtually illuminated by hand gestures captured through a LeapMotion controller, creating an MR scene.

The applications provided means of exemplifying the main concepts of the prototype platform. Throughout the article, several additions and modifications to the main application were presented in order to exemplify concepts in detail, such as the connectivity and interaction patterns achieved through the NFC tags and the feedforward and feedback formula-based, decision-based, and general processing nodes.

The conceptual framework was evaluated by letting two groups of students understand the underlying templates of designing the processing nodes and the flow of information within the two applications, followed by letting programmer and non-programmer students design configurations for other application ideas. The students also rated the utility, pleasantness, and innovation of the two applications, and commented on which features of a platform would encourage engagement as creators and explorers in order to provide sustainability in the context of digital transformation.

We conclude that the conceptual prototype platform is a foundation stone for creating multimodal distributed applications for an intelligent XR environment. Future research directions include finding ways to secure information transactions and enable micropayments with cryptocurrencies and devising a method for proving ownership of resources and developments in order to create a sustainable digital-platform environment.

Author Contributions: Conceptualization, E.B. and G.N.; Data curation, E.B.; Formal analysis, E.B.; Funding acquisition, D.-M.P. and G.N.; Investigation, E.B. and G.N.; Methodology, E.B. and G.N.; Project administration, D.-M.P.; Resources, E.B., G.N. and D.-M.P.; Software, E.B.; Supervision, D.-M.P.; Validation, E.B.; Visualization, E.B.; Writing – original draft, E.B. and G.N.; Writing – review & editing, E.B. and D.-M.P. All authors have read and agreed to the published version of the manuscript.

Funding: This research was partially supported by the Romanian Ministry of Research and Innovation, CCCDI-UEFISCDI, project number PN-III-P1-1.2-PCCDI-2017-0917, contract no. 21PCCDI/2018, within PNCDI III, project P2, "Efficient communications based on smart devices for in-car augmented reality interactive applications". The APC was funded through the IOAP of the University of Bucharest.

Data Availability Statement: No new data were created or analyzed in this study other than the quantitative data presented in Sections 5.3 and 6.2, and qualitative data presented in Section 6.1. Data sharing is not applicable to this article.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Parmentola, A.; Petrillo, A.; Tutore, I.; De Felice, F. Is blockchain able to enhance environmental sustainability? A systematic review and research agenda from the perspective of Sustainable Development Goals (SDGs). *Bus. Strategy Environ.* **2022**, *31*, 194–217. [CrossRef]
2. Anshari, M.; Syafrudin, M.; Fitriyani, N.L. Fourth Industrial Revolution between Knowledge Management and Digital Humanities. *Information* **2022**, *13*, 292. [CrossRef]
3. Van Dijk, J. *The Digital Divide*; John Wiley & Sons: Hoboken, NJ, USA, 2020.
4. Bran, E.; Bautu, E.; Popovici, D.M. Open Affordable Mixed Reality: A Manifesto. In Proceedings of the 2020 International Conference on Development and Application Systems (DAS), Suceava, Romania, 21–23 May 2020; pp. 177–184. [CrossRef]
5. Bran, E.; Bautu, E.; Popovici, D.M. Towards a sustainable future: Ubiquitous knowledge mixed reality museum. *Procedia Comput. Sci.* **2020**, *176*, 2878–2885. [CrossRef]
6. Bran, E.; Bautu, E.; Sburlan, D.F.; Puchianu, C.M.; Popovici, D.M. Ubiquitous Computing: Driving in the Intelligent Environment. *Mathematics* **2020**, *9*, 2649. [CrossRef]
7. Nadoleanu, G.; Staiculescu, A.R.; Bran, E. The Multifaceted Challenges of the Digital Transformation: Creating a Sustainable Society. *Postmod. Open.* **2022**, *13*, 300–316. [CrossRef] [PubMed]
8. Fuster Morell, M.; Espelt, R.; Renau Cano, M. Sustainable platform economy: Connections with the sustainable development goals. *Sustainability* **2020**, *12*, 7640. [CrossRef]
9. De Filippi, P.; Mannan, M.; Reijers, W. Blockchain as a confidence machine: The problem of trust & challenges of governance. *Technol. Soc.* **2020**, *62*, 101284. [CrossRef]
10. Nair, M.M.; Tyagi, A.K.; Sreenath, N. The future with industry 4.0 at the core of society 5.0: Open issues, future opportunities and challenges. In Proceedings of the 2021 International Conference on Computer Communication and Informatics (ICCCI), Coimbatore, India, 27–29 January 2021; pp. 1–7. [CrossRef]
11. Sharma, B.; Obaidat, M.S. Comparative analysis of IoT based products, technology and integration of IoT with cloud computing. *IET Netw.* **2020**, *9*, 43–47. [CrossRef]
12. Kim, J.C.; Laine, T.H.; Åhlund, C. Multimodal interaction systems based on internet of things and augmented reality: A systematic literature review. *Appl. Sci.* **2021**, *11*, 1738. [CrossRef]
13. Sendari, S.; Firmansah, A.; Aripriharta. Performance analysis of augmented reality based on vuforia using 3d marker detection. In Proceedings of the 2020 4th International Conference on Vocational Education and Training (ICOVET), Malang, Indonesia, 19 September 2020; pp. 294–298. [CrossRef]
14. Joshi, A.V. Amazon's machine learning toolkit: Sagemaker. In *Machine Learning and Artificial Intelligence*; Springer: Cham, Switzerland, 2020; pp. 233–243. [CrossRef]
15. Torres, D.; Dias, J.P.; Restivo, A.; Ferreira, H.S. Real-time feedback in node-red for iot development: An empirical study. In Proceedings of the 2020 IEEE/ACM 24th International Symposium on Distributed Simulation and Real Time Applications (DS-RT), Prague, Czech Republic, 14–16 September 2020; pp. 1–8. [CrossRef]
16. Liu, L.; Bahrami, M.; Chen, W.P. Automatic generation of ifttt mashup infrastructures. In Proceedings of the 2020 35th IEEE/ACM International Conference on Automated Software Engineering (ASE), Melbourne, VIC, Australia, 21–25 September 2020; pp. 1179–1183.
17. Kim, J. Advertising in the Metaverse: Research agenda. *J. Interact. Advert.* **2021**, *21*, 141–144. [CrossRef]
18. Tanwar, S. Blockchain Revolution from 1.0 to 5.0: Technological Perspective. In *Blockchain Technology*; Springer: Singapore, 2022; pp. 43–61. [CrossRef]
19. Kubek, M.; Unger, H. WebEngine Version 1.0: Building a Decentralised Web Search Engine. In *The Autonomous Web*; Springer: Cham, Switzerland, 2022; pp. 35–49. [CrossRef]
20. Wang, Q.; Li, R.; Wang, Q.; Chen, S. Non-fungible token (NFT): Overview, evaluation, opportunities and challenges. *arXiv* **2021**, arXiv:2105.07447.
21. Hirniak, J. Research-Driven Cardano DEX White Paper v1. 2021. Available online: https://docs.maladex.com/whitepaper.pdf (accessed on 19 July 2022).
22. Vallas, S.; Schor, J.B. What do platforms do? Understanding the gig economy. *Annu. Rev. Sociol.* **2020**, *46*, 273–294. [CrossRef]
23. Comunità, M.; Gerino, A.; Lim, V.; Picinali, L. Design and evaluation of a web-and mobile-based binaural audio platform for cultural heritage. *Appl. Sci.* **2021**, *11*, 1540. [CrossRef]
24. Xu, M.; Ng, W.C.; Lim, W.Y.B.; Kang, J.; Xiong, Z.; Niyato, D.; Yang, Q.; Shen, X.S.; Miao, C. A full dive into realizing the edge-enabled metaverse: Visions, enabling technologies, and challenges. *arXiv* **2020**, arXiv:2203.05471.
25. Van Zanten, J.A.; Van Tulder, R. Beyond COVID-19: Applying "SDG logics" for resilient transformations. *J. Int. Bus. Policy* **2020**, *3*, 451–464. [CrossRef]

26. Maavak, M. Horizon 2030: Will emerging risks unravel our global systems? *Salus J.* **2021**, *9*, 2–17.
27. Kovács, T.Z.; Bittner, B.; Huzsvai, L.; Nábrádi, A. Convergence and the Matthew Effect in the European Union Based on the DESI Index. *Mathematics* **2022**, *10*, 613. [CrossRef]
28. McLennan, M. *The Global Risks Report 2021*, 16th ed.; World Economic Forum: Cologny, Switzerland, 2021; Available online: https://www3.weforum.org/docs/WEF_The_Global_Risks_Report_2021.pdf (accessed on 18 July 2022).
29. Aly, H. Digital transformation, development and productivity in developing countries: Is artificial intelligence a curse or a blessing? *Rev. Econ. Political Sci.* **2020**. [CrossRef]
30. Li, Z.N.; Drew, M.S.; Liu, J. Augmented Reality and Virtual Reality. In *Fundamentals of Multimedia*; Springer: Cham, Switzerland, 2021; pp. 737–761. [CrossRef]
31. Jones, S.; Dawkins, S. The sensorama revisited: Evaluating the application of multi-sensory input on the sense of presence in 360-degree immersive film in virtual reality. In *Augmented Reality and Virtual Reality*; Springer: Cham, Switzerland, 2018; pp. 183–197. [CrossRef]
32. Kaivo-oja, J.; Lauraeus, T.; Knudsen, M.S. Picking the ICT technology winners-longitudinal analysis of 21st century technologies based on the Gartner hype cycle 2008–2017: Trends, tendencies, and weak signals. *Int. J. Web Eng. Technol.* **2020**, *15*, 216–264.
33. Rose, M. The immersive turn: Hype and hope in the emergence of virtual reality as a nonfiction platform. *Stud. Doc. Film.* **2018**, *12*, 132–149. [CrossRef]
34. Weiser, M. The computer for the 21st century. *ACM SIGMOBILE Mob. Comput. Commun. Rev.* **1999**, *3*, 3–11. [CrossRef]
35. Bhattacharyya, S.S.; Kumar, S. Study of deployment of "low code no code" applications toward improving digitization of supply chain management. *J. Sci. Technol. Policy Manag.* **2021**. [CrossRef]
36. Upadhyay, A.; Mukhuty, S.; Kumar, V.; Kazancoglu, Y. Blockchain technology and the circular economy: Implications for sustainability and social responsibility. *J. Clean. Production.* **2021**, *293*, 126130. [CrossRef]
37. Javed, F.; Antevski, K.; Mangues-Bafalluy, J.; Giupponi, L.; Bernardos, C.J. Distributed Ledger Technologies for Network Slicing: A Survey. *IEEE Access* **2022**, *10*, 19412–19442. [CrossRef]
38. Valadares, D.C.G.; Will, N.C.; Caminha, J.; Perkusich, M.B.; Perkusich, A.; Gorgônio, K.C. Systematic literature review on the use of trusted execution environments to protect cloud/fog-based Internet of Things applications. *IEEE Access* **2021**, *9*, 80953–80969. [CrossRef]
39. Pilkington, M. Blockchain technology: Principles and applications. In *Research Handbook on Digital Transformations*; Edward Elgar Publishing: London, UK, 2016. [CrossRef]
40. Curran, G.; Gibson, M. WikiLeaks, anarchism and technologies of dissent. *Antipode* **2013**, *45*, 294–314. [CrossRef]
41. Saran, S. Technology: Digital epiphany? COVID-19 and our tech futures. In *Challenges and Opportunities in the Post-COVID-19 World*; World Economic Forum: Cologny, Switzerland, 2020; pp. 24–27. Available online: https://www3.weforum.org/docs/WEF_Challenges_and_Opportunities_Post_COVID_19.pdf (accessed on 19 July 2022).
42. Hanjra, M.A.; Qureshi, M.E. Global water crisis and future food security in an era of climate change. *Food Policy* **2010**, *35*, 365–377. [CrossRef]
43. Webster, K. A circular economy is about the economy. *Circ. Econ. Sustain.* **2021**, *1*, 115–126. [CrossRef]
44. Türkeli, S.; Schophuizen, M. Decomposing the complexity of value: Integration of digital transformation of education with circular economy transition. *Soc. Sci.* **2019**, *8*, 243. [CrossRef]
45. Semanjski, I.; Lopez Aguirre, A.J.; De Mol, J.; Gautama, S. Policy 2.0 platform for mobile sensing and incentivized targeted shifts in mobility behavior. *Sensors* **2016**, *16*, 1035. [CrossRef]
46. Bada, A.O.; Damianou, A.; Angelopoulos, C.M.; Katos, V. Towards a green blockchain: A review of consensus mechanisms and their energy consumption. In Proceedings of the 17th International Conference on Distributed Computing in Sensor Systems (DCOSS), Pafos, Cyprus, 14–16 July 2021; pp. 503–511. [CrossRef]
47. Dann, D.; Peukert, C.; Martin, C.; Weinhardt, C.; Hawlitschek, F. Blockchain and Trust in the Platform Economy: The Case of Peer-to-Peer Sharing. *Wirtschaftsinformatik* **2020**, 1459–1473. [CrossRef]
48. Manda, M.I.; Ben Dhaou, S. Responding to the challenges and opportunities in the 4th Industrial revolution in developing countries. In Proceedings of the 12th International Conference on Theory and Practice of Electronic Governance, Melbourne, VIC, Australia, 3–5 April 2019; pp. 244–253. [CrossRef]
49. Bernal-Cárdenas, C.; Moran, K.; Tufano, M.; Liu, Z.; Nan, L.; Shi, Z.; Poshyvanyk, D. Guigle: A gui search engine for android apps. In Proceedings of the 2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion), Montreal, QC, Canada, 25–31 May 2019; pp. 71–74. [CrossRef]
50. Xu, C.; Wang, Y.; Quan, W.; Yang, H. Multi-person collaborative interaction algorithm and application based on HoloLens. In *Recent Trends in Intelligent Computing, Communication and Devices*; Springer: Singapore, 2020; pp. 303–315. [CrossRef]
51. Yang, A.M.; Li, S.S.; Ren, C.H.; Liu, H.X.; Han, Y.; Liu, L. Situational awareness system in the smart campus. *IEEE Access* **2018**, *6*, 63976–63986. [CrossRef]
52. Fatima, S.A.; Hussain, N.; Balouch, A.; Rustam, I.; Saleem, M.; Asif, M. IoT enabled smart monitoring of coronavirus empowered with fuzzy inference system. *Int. J. Adv. Res. Ideas Innov. Technol.* **2020**, *6*, 188–194.
53. Butakova, M.A.; Chernov, A.V.; Guda, A.N.; Vereskun, V.D.; Kartashov, O.O. Knowledge representation method for intelligent situation awareness system design. In *International Conference on Intelligent Information Technologies for Industry*; Springer: Cham, Switzerland, 2018; pp. 225–235. [CrossRef]

54. Gajdošík, T. Towards a conceptual model of intelligent information system for smart tourism destinations. In *Software Engineering and Algorithms in Intelligent Systems*; Springer: Cham, Switzerland, 2018; pp. 66–74. [CrossRef]
55. Wilson, K.B.; Karg, A.; Ghaderi, H. Prospecting non-fungible tokens in the digital economy: Stakeholders and ecosystem, risk and opportunity. *Bus. Horiz.* **2021**, *65*, 657–670. [CrossRef]
56. Mercan, S.; Kurt, A.; Akkaya, K.; Erdin, E. Cryptocurrency solutions to enable micropayments in consumer IoT. *IEEE Consum. Electron. Mag.* **2021**, *11*, 97–103. [CrossRef]
57. Bran, E.; Bautu, E.; Popovici, D.M.; Braga, V.; Cojuhari, I. Cultural Heritage Interactive Dissemination through Natural Interaction. In Proceedings of the 16th RoCHI ACM Conference, Bucharest, Romania, 17–18 October 2019; pp. 156–161.